MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Schöpe Till**	Personal ID number:	**466047**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Development and Evaluation of a Concept for the Augmentation of Data to Train Neuronal Networks for Semantic Segmentation of LiDAR-Pointclouds

Master’s thesis title in Czech:

Vývoj a zhodnocení konceptu pro rozšíření dat pro trénování neuronových sítí pro semantickou segmentaci LiDAR-Pointclouds

Guidelines:

- Develop and Evaluate a Concept for Augmentation of LiDAR-pointclouds
- Find a network-architecture that is suitable for the evaluation of different augmentation techniques
- Find a proper pointcloud-to-input transformation so that LiDAR-pointclouds can be fed to the neural-network

Bibliography / sources:

[1] X. YUE et al.: A LiDAR Point Cloud Generator: from a VirtualWorld to Autonomous Driving, arXiv:1804.00103 [cs] (Mar. 30, 2018), arXiv: 1804.00103
[2] A. GARCIA-GARCIA et al.: A Review on Deep Learning Techniques Applied to Semantic Segmentation, arXiv:1704.06857 [cs] (Apr. 22, 2017), arXiv: 1704.06857
[3] P. SIMARD, D. STEINKRAUS, and J. PLATT: Best practices for convolutional neural networks applied to visual document analysis, Seventh International Conference on Document Analysis and Recognition, 2003.Proceedings. Seventh International Conference on Document Analysis and Recognition, vol. 1, Edinburgh, UK: IEEE Comput. Soc, 2003, pp. 958–963

Name and workplace of master’s thesis supervisor:

Bastian Lampe, M.Sc., RWTH Aachen University

Name and workplace of second master’s thesis supervisor or consultant:

Ing. Martin Hlinovský, Ph.D., Department of Control Engineering, FEE

Date of master’s thesis assignment: **05.02.2019** Deadline for master's thesis submission: **07.06.2019**

Assignment valid until: **30.09.2020**

Bastian Lampe, M.Sc.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Zentrales Prüfungsamt/Central Examination Office

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Name, Vorname/Last Name, First Name	Matrikelnummer (freiwillige Angabe) Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Ort, Datum/City, Date	Unterschrift/Signature

*Nichtzutreffendes bitte streichen
*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.
§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Ort, Datum/City, Date	Unterschrift/Signature

Abstract

An accurate environment perception is a key requirement for automated vehicles. To help fulfill this requirement, a promising approach is the semantic segmentation of LiDAR-pointclouds using neural networks. The acquisition of large amounts of annotated data needed for training these networks is challenging due to the high cost of manually labeling pointclouds.

In this work, data augmentation is proposed to obtain a high number of annotated LiDAR-pointclouds without the need of manual labeling. Two augmentation strategies are presented, the first one being the creation of semi-artificial samples and the second one the application of label-preserving transformations. Semi-artificial samples are created by automatically extracting objects from a controlled environment and inserting them into scenes where these objects do not occur.

The results suggest that semi-artificial pointclouds can be successfully used as a supplement or as a substitution of real data. An intersection over union of 31.7 % can be achieved on a data set of real world scenes when only semi-artificial pointclouds are used for training. Additionally, the recall on the real world data set can be increased if the semi-artificial data set is further augmented with label-preserving transformations.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
BP	Backpropagation
CNN	Convolutional Neural Network
DNN	Deep Neural Network
ELU	Exponential Linear Unit
GD	Gradient Descent
GPU	Graphics Processing Unit
IKA	Institute for Automotive Engineering
IoU	Intersection over Union
leakyReLU	Leaky Rectified Linear Unit
LiDAR	Light Detection and Ranging
MEMS	Micro-Electro-Mechanical System
MSE	Mean Square Error
pp	percentage points
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
VRU	Vulnerable Road Users

1 Introduction

The German Federal Statistical Office registered 3180 fatalities on German roads in 2017. 88% of all registered accidents can be attributed to human errors. Counting 1507, almost half of all fatalities affect Vulnerable Road Users (VRU), such as pedestrians or bikers. [STA18] Automated vehicles bear the potential to drastically reduce the number of road accidents [SHL18]. Worldwide, numerous research institutions and automotive companies devote their research to automated driving.

An accurate environment representation is fundamental for many automated driving applications. The ability of an automated vehicle to sense its surroundings is the basis for behavior generation and trajectory planning. In order to achieve appropriate environment perception, it is often not sufficient to know the location of surrounding objects, but also to classify them. A tree on the roadside can require a different behavior than a pedestrian about to walk across the road.

Sensors based on Light Detection and Ranging (LiDAR) represent a promising supplement to cameras for environment perception as they are independent of ambient light and able to provide depth information. A LiDAR-sensor sends out laser beams which are reflected by surrounding objects. When detecting a reflected laser beam, either time-of-flight or difference in wavelength can be used to compute the distance between the sensor and the object. The points of reflection are accumulated in a pointcloud, enabling a 3-dimensional measurement of the sensor's surroundings.

Semantic segmentation performs classification on point-level. Typically, it is performed on images or pointclouds, i.e. a class is assigned to every pixel or every point respectivly. The method is denoted as one of the key computer vision challenges as it represents the basis for full scene understanding [GAR17]. It is already used in various applications in automated driving [ESS09; GEI12; COR16]. Performing semantic segmentation on LiDAR-pointclouds can therefore play an important role in creating an accurate environment model.

With the recent rise of deep learning, deep convolutional neural networks have become a popular choice in the whole area of computer vision [LEC15], including semantic segmentation. These networks prove to have higher accuracy than previous approaches and are better at correctly processing new data [GAR17]. The decreasing cost of Graphics Processing Units (GPUs) enables the use of deep learning techniques in automated driving, where large amounts of data need to be processed in a short amount of time. The availability of sufficient training data is crucial for the performance of a neural network. Larger amounts of labeled network inputs typically lead to a better performance. Labeling inputs by hand is prone to errors and is both time and cost intensive.

Data augmentation is the process of generating additional network-inputs by alterations of existing data. They can be used to enlarge a data set without the need for additional labeling.

Exemplary augmentation techniques are rotations or the addition of Gaussian noise. For example, rotating the pointcloud of a scene changes the location of points, but the class assigned to each point stays the same.

Content and Structure of this Work

This master thesis deals with the challenge of automatically creating large labeled pointcloud data sets while keeping the manual labeling effort as low as possible. A concept to create augmented data is developed. For evaluation, the performance of a neural-network for semantic segmentation trained on the generated data sets is measured. Therefore, a suitable network topology for semantic segmentation of pointclouds has to be found. Convolutional Neural Networks (CNNs) represent a popular choice in semantic segmentation tasks and are therefore focused. The pointclouds have to be transformed into a format that can serve as input for the neural network.

First, state of the art research regarding the topics worked on in this thesis is presented in chapter 2. This includes a review about current LiDAR-Technology, an introduction to artificial neural networks as well as a presentation of the most recent progress in data augmentation. Chapter 3 describes the motivation and derives a research question to study. The concept developed in this thesis is presented in chapter 4. Furthermore, experiments for evaluation and the presentation of their results can be found in chapters 5 and 6. A final conclusion and an outlook are elaborated in chapter 7.

2 State of the Art

This chapter presents the current state of the art in data augmentation for semantic segmentation of LiDAR pointclouds using neural networks. First, current LiDAR technology is reviewed. Then, a brief introduction to the fundamentals of machine learning and artificial neural networks is given. Furthermore, recent work about the topics treated in this thesis is presented. This includes data labeling and augmentation, semantic segmentation and the transformations of pointclouds to matrices.

2.1 Light Detection and Ranging

The semantic segmentation task examined in this thesis is performed on pointclouds. Those pointclouds originate from LiDAR measurements. The LiDAR principle is based on a laser diode sending out pulses which can get reflected by targets. The reflected laser enables computation of the distance traveled by the laser pulse. Time of flight or phase-shift can be used as the foundation. Considering the time-of-flight sensors used in this work, using the speed of light c, the time the pulse is generated t_1 and the time the pulse is received t_2, the reflection distance r can be calculated as follows:

$$r \approx c \cdot \frac{t_2 - t_1}{2}, \quad if \;\; c >> v \qquad \text{Eq. 2-1}$$

The movements of the sensor and the reflecting object during the measurement, e.g. if mounted on a moving vehicle, are neglected as the speed of light is much larger in comparison to a vehicle's velocity.

By sweeping the laser beam, multiple points within a single plane can be collected.Multiple laser diodes can be stacked with increasing elevation angles to receive a 3D-pointcloud representing the sensors environment. Often, the intensity I of the reflected laser beam is measured in addition to the spatial information. The principle is depicted in figure 2-1. The target coordinates are typically represented as spherical coordinates. The zenith angle Θ corresponds to the laser diode's elevation angle and the azimuth φ to the sweeping angle. A representation as Cartesian coordinates can be obtained by applying following transformation:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = r \cdot \begin{pmatrix} \sin\Theta \cdot \cos\varphi \\ \sin\Theta \cdot \sin\varphi \\ \cos\Theta \end{pmatrix} \qquad \text{Eq. 2-2}$$

There are two different approaches used to sweep the laser beams: The first one realizes the sweeping by rotating the whole sensor around itself. By doing so, a full 360° measurement can be performed. The latter approach does not move the sensor itself but deflects the laser beam, for example by using a movable mirror [VEL07]. Those sensors can be produced smaller and cheaper, though they do not provide a 360° measurement. Recent research tries to realize the deflection via Micro-Electro-Mechanical System (MEMS) in order to further reduce size and production cost [PAR18]. As sensors produced this way do not have any freely moving parts, they are often referred to as Solid-State-LiDAR sensors.

As mentioned in chapter 1, LiDAR sensors are a promising supplement to cameras. The sensor works independently of ambient light and enables an accurate measurements even in long distances and up to 360° around the vehicle. However, similar to cameras, objects can be occluded. An object reflecting a laser pulse occludes all further obstacles in the laser's possible ray behind that object. Therefore, heavy weather conditions are challenging for LiDAR sensors. A high number of raindrops or snowflakes might prevent that a laser pulse is reflected by an obstacle, leading to a high number of occlusions and noise.

Alike the automated driving research, LiDAR development is advancing rapidly. 360° LiDAR-sensors are widely used for fully automated driving research. [PAR18] offers a good state-of-the-art market survey about LiDAR manufacturers. Current sensors, for example as produced by Velodyne, can evaluate up to 700.000 measurement points per second with a rotation rate of up to 20 Hz. The azimuth resolution typically is few tenths of degrees. Velodyne offers sensors with 16, 32 or 64 laser diodes stacked, a 128 version is in the release. However, according to [PAR18], the only sensor currently used in series consumer vehicles is based on the deflecting mirror principle and offers a field of view of 145°. Released in 2017 by Valeo, it has been implemented in the Audi A8 as a part of the traffic jam assist.

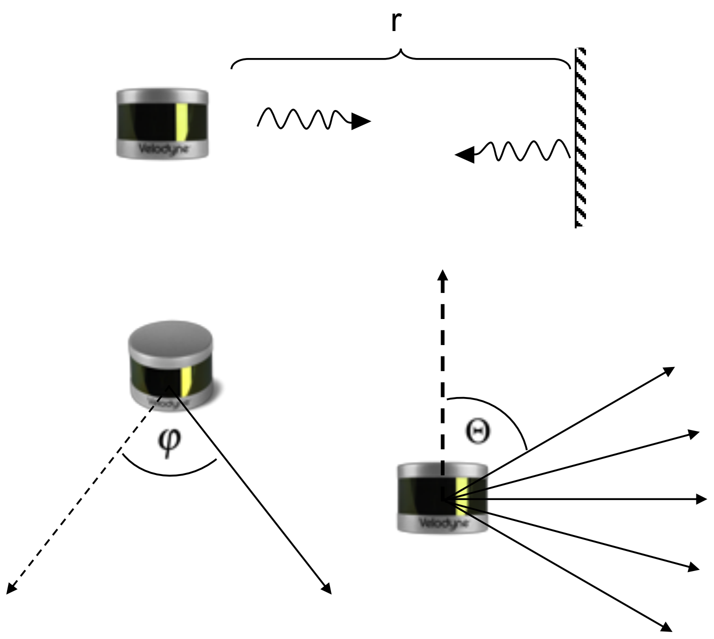


Fig. 2-1: LiDAR measurement principle. The distance is computed by a time-of-flight-measurement. The point of reflection can be represented in spherical coordinates, with the radius corresponding to the distance r, the azimuth φ to the sweeping angle and the zenith θ to the laser's elevation angle. Image of the sensor taken from www.velodyne.com.

2.2 Artificial Neural Networks

As mentioned in chapter 1, neural networks are widely used for semantic segmentation tasks. As this thesis deals with the augmentation of pointclouds to train neural networks, a brief introduction to Artificial Neural Networks (ANNs) is given in this section.

ANNs are mathematical functions inspired by the human brain [ROS58]. In analogy to its biological counterpart, the basic building blocks of each neural network are called (artificial) neurons. A single neuron takes an input vector and processes it into an output scalar. First, a transfer function computes the weighted sum of the inputs, resulting in the signal z. Together with a bias b, this signal is then fed to an activation function which computes the neuron's output. The output is called activation a. The activation functions of a neuron of an ANN need to be nonlinear in order for the ANN to model a nonlinear function.. The biological inspiration behind an activation function is that a biological neuron *fires* above a certain input threshold and stays inactive below it [ROS58].

Typical activation functions are step-function, sigmoid-function, tangens hyperbolicus, Rectified Linear Unit (ReLU), Leaky Rectified Linear Unit (leakyReLU) or Exponential Linear Unit (ELU) [MEH18]. For a better understanding, they are plotted in figure 2-2.

Summarizing the neuron i in a single function, the activation a can be expressed as follows:

$$a_i = f\left(\begin{pmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ w_{m,i} \end{pmatrix}^T \cdot \begin{pmatrix} x_{1,i} \\ x_{2,i} \\ \vdots \\ x_{m,i} \end{pmatrix} + b_i\right) = f\left(\mathbf{w_i}^T \cdot \mathbf{x_i} + b_i\right) := f(\mathbf{z_i}, b_i) \qquad \text{Eq. 2-3}$$

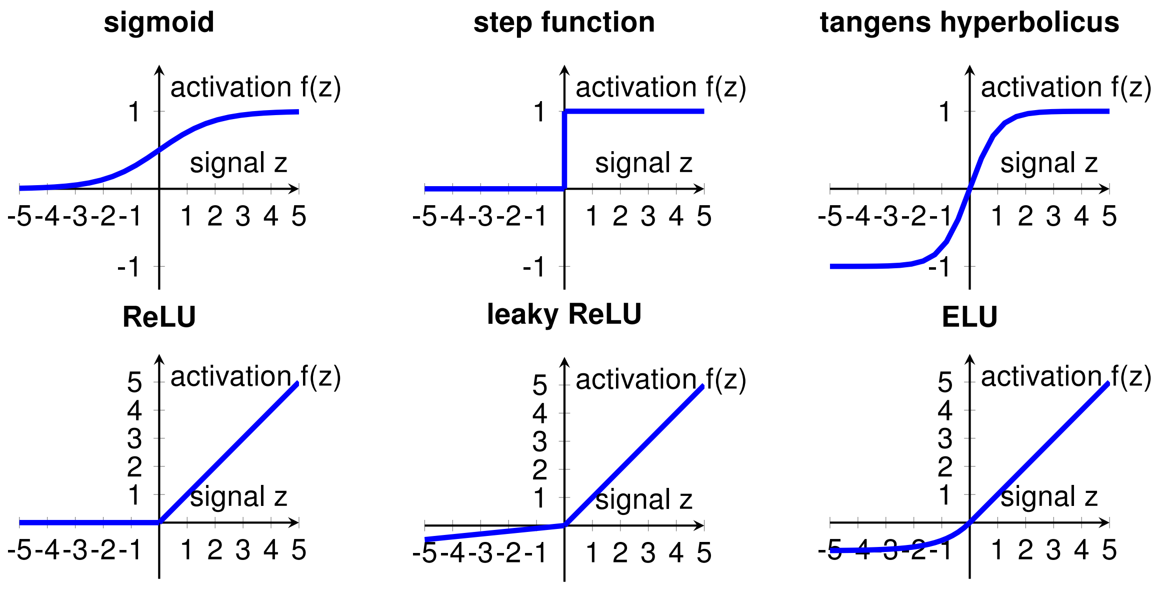


Fig. 2-2: Typical activation functions for artificial neurons

A graphical representation of an artificial neuron can be seen in figure 2-3. Multiple neurons with individual weights and biases can be combined to form a layer. A neural network is a combination of multiple neurons forming one or more layers. If multiple layers exist, the outputs of a layer serve as input for the following one. This way, the output of a neural network is a chained combination of the networks layers. The first layer of a neural network is called input layer, the last one is called output layer. Often, the output is also known as the prediction. Possible layers between input and output layers are denoted as hidden layers. If every neuron of a layer is connected to all neurons of the following layer, the former layer is called a fully connected layer. An illustration is depicted in figure 2-4.

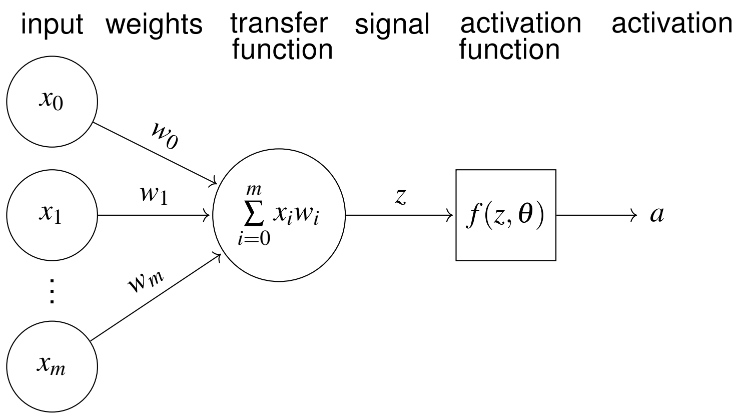


Fig. 2-3: Illustration of an artificial neuron. The signal, i.e. the weighted sum of all inputs, is fed to a nonlinear function creating the neuron's output.

The output obtained through a certain input combination can be influenced by modifying the neurons' weights and biases. By this modification, the neuron can be used for *learning*. A model is said to learn if it improves its performance measure *P* with respect to a task *T* with increasing experience *E* [MIT97]. The experience can appear in the form of data for example. By using data to modify weights, a neural network can learn a large amount of different functions to solve tasks like regression or classification. The phase of modifying weights is called *training*. Receiving outputs for new, unseen inputs is typically called *inference*. A more detailed presentation of training neural networks is found in section 2.4 and 2.5.

If a neural network is trained to perform classification tasks, typically each class is represented by its own output neuron. In an ideal case, the neuron assigned to the correct class shows an output while all the other neurons stay inactive at zero. However, this is usually not the case and all outputs show some kind of activity. In most cases, the class belonging to the neuron with the maximum output is chosen. In order to receive a measure of how good the networks classification is, a *softmax* function can be applied to the networks outputs. *Softmax* is a normalized exponential function. It normalizes all outputs of a network to positive values between zero and one. It is therefore used to represent probabilities, i.e. the certainty with which the neural network predicts its own classification. Considering K output neurons, computing the *softmax* for the activation $a_{out,n}$ of output neuron n looks as follows: [GOO16]

$$softmax(a_{out,n}) = \frac{e^{a_{out,n}}}{\sum_i^K e^{z_i}} \qquad \text{Eq. 2-4}$$

For neural networks, the term width is used to describe the number of neurons per layer, the term depth refers to the total number of layers. Therefore Deep Neural Networks (DNNs) are networks with a high number of hidden layers. Similarly, deep learning referes to the development and training of DNNs.

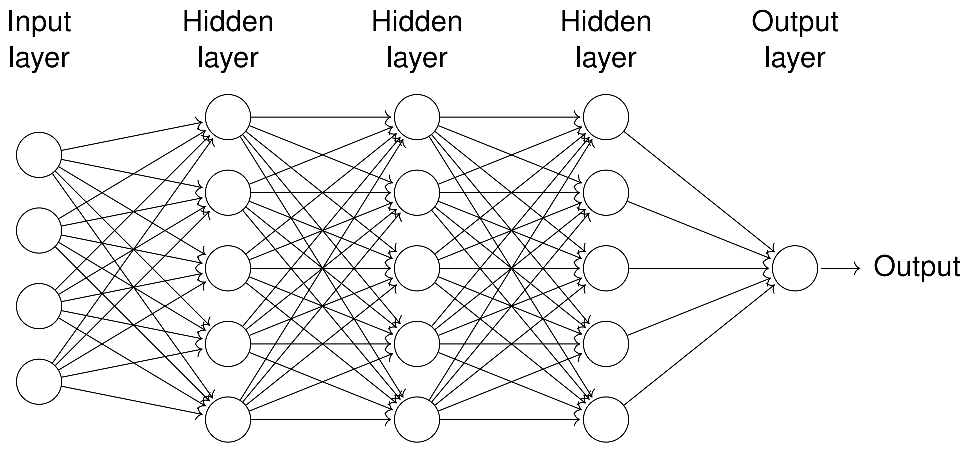


Fig. 2-4: Visualization of a Neural Network consisting of three hidden layers.

2.3 Convolutional Neural Networks

CNNs are neural networks specialized on matrix-like inputs, for example images. As their name suggests, their operation is based on the mathematical convolution [GOO16]. Fully connected layers connect every input to every output of a layer with an individual weight. The number of trainable parameters increases rapidly with additional neurons and layers. Even with today's computation power, training networks with many layers can therefore quickly become unfeasible with respect to time and memory. CNNs reduce the number of total weights by introducing sparse interactions and weight sharing [LEC89; GOO16].

Sparse interactions abolish the interaction of each input neuron with each output neuron, as done in fully connected layers. Instead, only a small subset of neighbouring inputs is considered by a neuron. This subset is referred to as kernel. For example, a 3x3 kernel applied to a single channel image takes a square of 3x3 pixels of an image as input to a neuron. Hence, a 3x3 kernel can process nine inputs and contains nine weights. This small square as input is often sufficient to detect small features, like edges for example. By applying the kernel to different locations of an image, for example by shifting it one pixel at a time, edges can be detected everywhere in the image. The advantage is the relatively small number of weights needed. In this example, only nine weights have to be trained, no matter how many pixels an input image contains.

Usually, multiple kernels are applied to the same input. Often, the weights in the first layers of a neural network lead to the detection of low level features like edges, while the latter layers use these features for a higher level detection, circles for example [GOO16].

The number of pixels the filter is shifted in each step is denoted as stride. Padding describes how to handle the outermost pixels. Typically, if the filter's center is located at an outermost pixel, the filter elements outside of the image are set to zero, namely zero-padding. Using zero-padding and a stride of one would result in an output of the exact shape of the input. Figure 2-5 visualizes this process. Setting a larger stride for example might be used to down-sample an image. A stride of two results in an output with quarter the resolution of the input.

The repeated use of the shifted kernel can be understood as a convolutional operation. A filter with kernel K applied to an image I produces a so called feature map F with pixel positions i, j. With respect to equation 2-3 the operation can be described by following formula: [GOO16]

$$F(i,j) = \sum_{m}^{k_x}\sum_{n}^{k_y} I(m,n) \cdot K(i-m, j-n) \qquad \text{Eq. 2-5}$$

Performing this operation on the whole Image thus results in a convolution of the filter kernel across the image, explaining the name origin of CNNs: [GOO16]

$$F(i,j) = (I * K)(i,j) \qquad \text{Eq. 2-6}$$

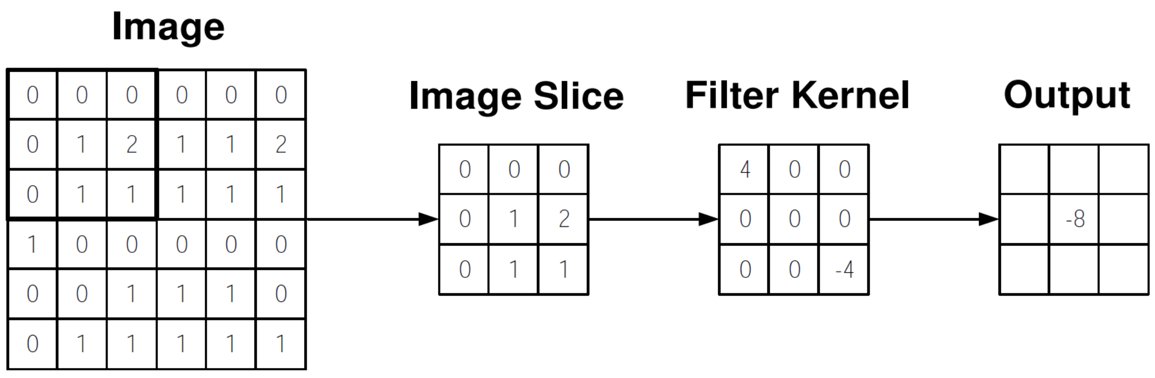


Fig. 2-5: A visual representation of a convolutional layer. The center element of the kernel is placed over the input vector and convolution is then calculated and replaced with a weighted sum of itself and any nearby pixels. Taken from [OSH15].

The previously described techniques are not limited to two dimensional inputs. A typical kernel has a size of $k_x \times k_y \times c$. Thereby, k_x and k_y describe the above mentioned pixel size of an input image. Additionally, c represents the depth of an image, for example 3 in an RGB-colored image or the height in a pointcloud. While k_x and k_y, c is therefore fixed by the depth of the grid-like input. The depth of the output is equal to the number of different kernels used by the layer.

Layers performing the previously described techniques are the main building block of CNNs. They are called convolution or convolutional layers. However, other layer types or a combination of layers types can be used as well. The two most common are pooling and deconvolution layers, which are explained in the following.

Pooling Layers

As mentioned in the precious section, typical convolutional layers produce an output matrix of the same width and height of its inputs, only alternating the number of channels. However, often a reduction of features to train is beneficial. For example, feature reduction plays an important role in overfitting prevention. Overfitting is presented in section 2.6. In order to reduce the number of features, i.e. the resolution in image processing, pooling layers can be used [WEN92]. They make use of a sliding window as well, but they do not contain any trainable weights. They map a defined number of inputs to one output. The most common pooling operations are average-pooling or max-pooling. As their names suggest, average pooling computes the average of the inputs and max-pooling forwards the maximum value [RIE99; OSH15]. A visualization of pooling is found in 2-6. Even though convolutional layers are able to downsample the inputs, pooling layers are a popular choice as they do not contain any weights leading to a larger model to train.

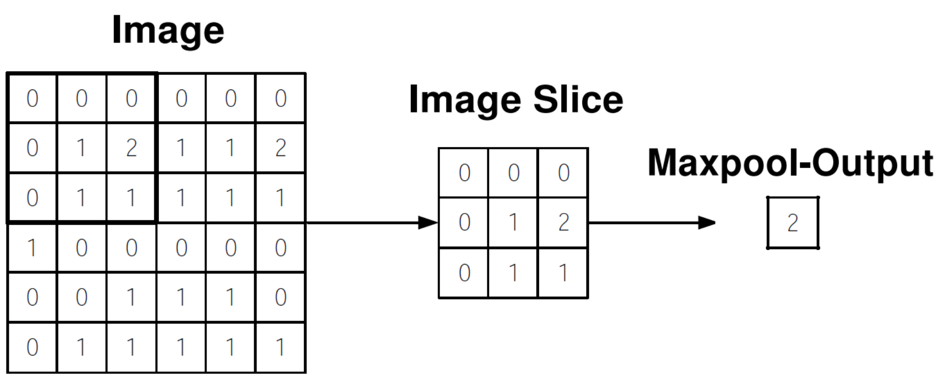


Fig. 2-6: A visual representation of a max-pooling layer. The pooling operation downsamples the image slice by forwarding the slice’s maximum value. No trainable weights are used. Modified from [OSH15].

Deconvolutional Layers

As described in the previous section, pooling layers can be used to perform a resolution reduction. In contrast to that, deconvolutional layers are used to increase the resolution. They take one pixel as input and use a filter with kernel size $k_x \times k_y$ to map it to multiple pixels in the following layer as visualized in figure 2-7. However, their name is misleading. The operation performed is not equivalent to the mathematical deconvolution.

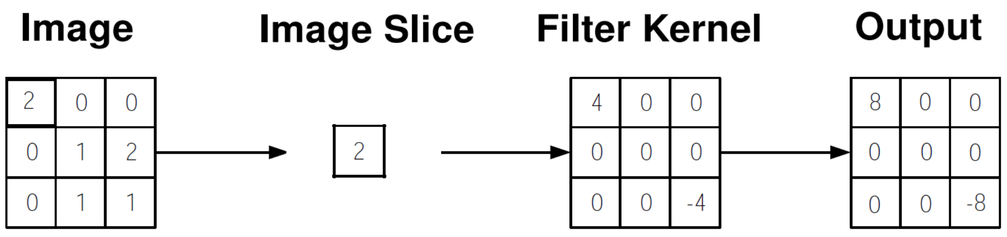


Fig. 2-7: A visual representation of a deconvolutional layer. When comparing it to figure 2-5, it can be seen that the layer performs the opposite operation compared to a convolutional layer. However, this operation does not equal the mathematical deconvolution. Modified from [OSH15].

Encoder-Decoder Structures

Encoder-Decoder neural networks are a popular design category for the development of neural network topologies. The key idea is to encode a large amount of inputs into a small number of features in the middle of the neural network. The features can then be decoded to the output. Encoder-Decoder networks are often used for semantic segmentation of images [BAD17]. The image is downsampled, possibly multiple times, to reduce the number of features. This part of the network is called encoder. The encoder is followed by the decoder, which increases the number of features by upsampling. Eventually, downsampling and following upsamling produce an output of the same height and width as the input image. By encoding the image into a small number of features, the neural network is forced to learn the most important features of an image. For facial recognition, this could be a detected nose, an eye or also features not recognizable by humans.

2.4 Taxonomy of Learning Tasks

Machine learning tasks can be broadly divided into three categories: Supervised learning, unsupervised learning and reinforcement learning.

In supervised learning, both the inputs and desired outputs are known during training. The known outputs are called ground truth. The machine learning model's output is compared to the ground truth using an error function [LEC15]. The models parameters can be modified in order to reduce the discrepancy between ground truth and prediction. Typical use-cases for supervised learning are classification or regression tasks. Typically, the weights are initialized randomly.

In unsupervised learning tasks, ground truth values are unknown or do not exist. The machine learning model learns to find correlations or patterns in the data set. The unsupervised learning task best known is clustering [MEH18].

Models trained by reinforcement learning adapt their parameters by receiving a reward for their output instead of a ground truth. An error function is not given in this type. During the training process, the models parameters are changed without any prior knowlegde how this modification might change the received reward. However, by comparing the rewards of different parameter variations, a model can be learned [SUT98].

2.5 Training Neural Networks

The semantic segmentation task covered in this thesis belongs to supervised leaning. Therefore, this section will focus on training of neural networks for supervised learning. As explained in the previous section, an error function is used to compare prediction and ground truth in order to adapt the networks weight.

Various possible error functions exist for different purposes. A typical error function for regression tasks is a simple Mean Square Error (MSE). It punishes the distance between prediction and ground truth. For the prediction $\hat{\mathbf{y}}$, the ground truth $\mathbf{y}$ and the numer of outputs n, MSE can be expressed as [MEH18]:

$$L_{MSE}(y,\hat{y}) = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2 \qquad \text{Eq. 2-7}$$

Cross entropy is a common error function used in classification tasks. The error L with prediction $\hat{y}$ and ground truth y can be described as:

$$L_{cross\ entropy}(y,\hat{y}) = \frac{1}{n}\sum_{i}^{n} y_i \cdot \log \hat{y}_i \qquad \text{Eq. 2-8}$$

A classification model typically outputs probabilities for possible classes, as explained when introducing *softmax* in section 2.2. The model therefore learns a probability distribution. Cross-entropy is a measure for comparing how similar two probability distributions are. It is therefore a good loss function to compare the ground-truth distribution with the network's output. [GOO16]

The most common technique to optimize the weights is called Gradient Descent (GD). The key idea of GD is to compute the gradient of the error function with respect to the weights and to descend the error function along this gradient. The factor to scale the weight changes is called learning rate λ. The simplest form of GD can be described as follows:

$$w_{t+1} = w_t - \lambda \cdot \frac{\partial L(y,\hat{y})}{\partial w_t} \qquad \text{Eq. 2-9}$$

Gradient computation and weight adaption are then performed on the whole data set. An iterative method to modify the weights is called Stochastic Gradient Descent (SGD). Instead of the whole data set, randomly selected samples are grouped into batches. The weight adaption is then performed on those batches iteratively. Iterating through all batches once is defined as an epoch and done repeatedly during training. Not performing GD on the whole data set leads to less memory demand at one time as the gradient for fewer samples has to be computed at the same time.

Calculating the gradient demands expensive computations as multiple layers lead to terms with long chains of weights. Today's most common approach for gradient computation is called (Error-)Backpropagation (BP). [RUM86] paved the way for using BP for the adaption of weights in neural networks. As BP is used successfully in most neural networks, it is used in this thesis as well. Therefore, the idea behind BP is briefly presented below.

Instead of computing the whole gradient at once, the error at the end of each layer is computed recursively from the last to the first layer. As the input and current weights are given, the signal z_i as well as the activation $a_i = f(z_i)$ is known for each neuron in the network.

As mentioned in section 2.2, the outputs of a layer l serve as inputs for its following layer $l+1$, with the exception of the output layer l_{out}. The weight vector connecting the layer l with neuron n of the following layer $l+1$ can be described as $\mathbf{w}_t^{l,n}$. For the sake of a better understanding, backpropagation is presented for a neural network with a single output trained using MSE. Regarding only the output layer $l = l_{out}$, the weight change $\Delta\mathbf{w}$ in equation 2-9 can be rearranged to the following formula, introducing the error signal δ:

$$\Delta\mathbf{w}_t^{l_{out},n} = -\lambda \cdot \frac{\partial L}{\partial \mathbf{w}_t^{l_{out},n}} \qquad \text{Eq. 2-10}$$

$$= -\lambda \cdot \frac{\partial L}{\partial a^{l_{out},n}} \cdot \frac{\partial a^{l_{out},n}}{\partial z^{l_{out},n}} \cdot \frac{\partial z^{l_{out},n}}{\partial \mathbf{w}_t^{l_{out},n}} \qquad \text{Eq. 2-11}$$

$$= -\lambda \cdot \delta^{l_{out},n} \cdot \frac{\partial z^{l_{out},n}}{\partial \mathbf{w}_t^{l_{out},n}} \qquad \text{Eq. 2-12}$$

Using the fact that the activation of a layer serves as input of the following layer, derivating the last factor of equation 2-12 leads to the following formula, for all layers:

$$\frac{\partial z^{l,n}}{\partial w_t^{l,n}} = \frac{\partial\left((w_t^{l,n})^T \cdot x^{l,n}\right)}{\partial w_t^{l,n}} = \mathbf{x}^{l,n} = \mathbf{a^{l-1}} \qquad \text{Eq. 2-13}$$

As mentioned before, the activations in the networks are all known by propagating the input to the output. Regarding equation 2-11, it can be seen that all variables needed for computing the error signal $\delta^{l_{out},n}$ for the neurons of the output layer are known. The loss function directly depends on the last layer's activation. Hence, the loss function and the activation function can be derived without further previous computations. For the sake of simplicity, an example is given using MSE as loss function:

$$\delta^{l_{out},n} = \frac{\partial L}{\partial a^{l_{out},n}} \cdot \frac{\partial a^{l_{out},n}}{\partial z^{l_{out},n}} \qquad \text{Eq. 2-14}$$

$$= \frac{\partial \left(\frac{1}{2}(y - a^{l_{out},n})^2\right)}{\partial a^{l_{out},n}} \cdot \frac{\partial a^{l_{out},n}}{\partial z^{l_{out},n}} \qquad \text{Eq. 2-15}$$

$$= -(y - a^{l_{out},n}) \cdot f'(z^{l_{out},n}) \qquad \text{Eq. 2-16}$$

The influence of activations from hidden layers on the error function is however more complicated. To receive $\delta^{l,n}$, the error signal can be propagated backwards through the network:

$$\delta^{l,n} = \frac{\partial L}{\partial a^{l,n}} \cdot \frac{\partial a^{l,n}}{\partial z^{l,n}} \qquad \text{Eq. 2-17}$$

$$= \left(\sum_{n_{next}} \frac{\partial L}{\partial a^{l+1,n_{next}}} \cdot \frac{\partial a^{l+1,n_{next}}}{\partial z^{l+1,n_{next}}} \cdot \frac{\partial z^{l+1,n_{next}}}{\partial a^{l+1,n}} \right) \cdot \frac{\partial a^{l,n}}{\partial z^{l,n}} \qquad \text{Eq. 2-18}$$

$$= \left(\sum_{n_{next}} \delta^{l+1,n_{next}} \cdot \mathbf{w}^{l+1,n_{next}} \right) \cdot \frac{\partial a^{l,n}}{\partial z^{l,n}} \qquad \text{Eq. 2-19}$$

In summary, BP can be described as follows: First, an input is passed through the neural network, leading to a computation and storage of all activations and the output. Afterwards, the error signal for the output layers is computed. This error signal can be used to modify the weights of the last layer. Additionally, the error signal is used for the computation of the preceding layer's error signal. By passing the error signal from the output layer to the input layer step by step, the partial derivatives of the error function with respect to all weights can be computed and used for the weight update.

2.6 Capacity, Under- and Overfitting

The previous section explained how weights can be modified in order to reduce the error the model produces on the training data. As a matter of fact, achieving low errors on the training data is usually not sufficient for a well trained neural network. The ability to perform well not only on the training data, but as well on new data is crucial. This ability is known as generalization. Performance on training and new data can be described by the two terms underfitting and overfitting. Not being able to achieve a low training error is known as underfitting [GOO16]. The state of performing well on training data but showing missing generalization capability is known as overfitting. [GOO16] offers a definition and shows strategies how to deal them during training. Therefore, it will be the basis of the following explanations.

Capacity describes a model's capability to learn complex prediction rules. A model of poor capacity might not be able to learn suitable prediction rules for its training data. However, increasing the capacity too far might enable the model to learn every input/output combination directly, including outliers and noise, instead of finding a proper rule. They therefore show a high bias towards the characteristics of the training data set. A capacity trade-off between under- and overfitting has to be found. This trade-off is visualized in figure 2-8.

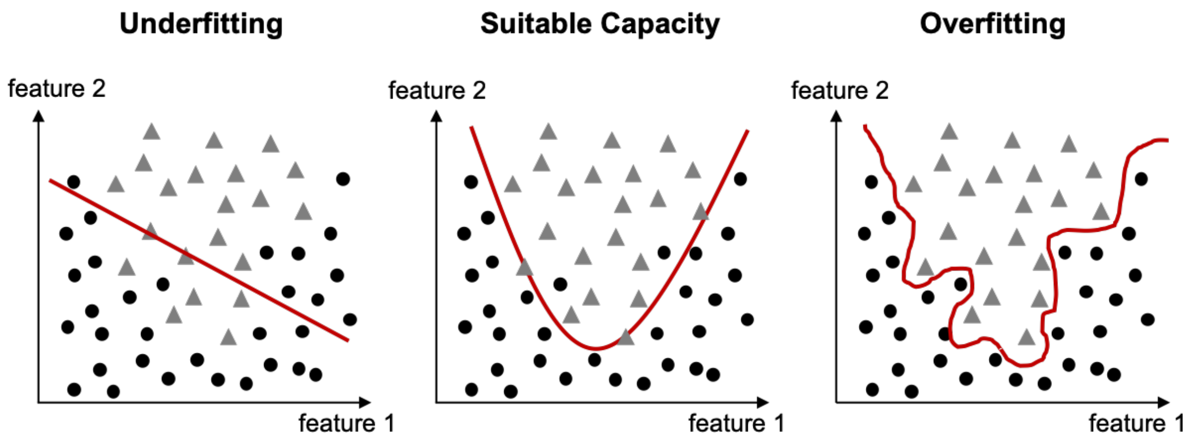


Fig. 2-8: Illustration of over- and underfitting with a model trained to separate circles and triangles.. A model underfitting has not enough capacity to separate the areas containing triangles from the ones containing circles. However, adding too much capacity leads to a perfect separation in this case, but from the plot it can be seen that the model learned to consider noise and outliers as well. A model of suitable capacity is able to find a decent separation border, accepting some errors due to noise or outliers.

In order to detect an over- and underfitting model, the available data set is typically divided into two parts. The first part is used for the purpose of training the neural network. The second part is never directly used for weight modification and can therefore be used to estimate the performance on new data. Due to their purposes, the former part is called training data set and the latter is called validation data set. After each epoch, the loss is computed for both the training and the validation samples. In the beginning of the training, both training and validation error should decrease each epoch. If, during the training process, the validation error begins to rise while the training error still keeps falling, the model starts overfitting.

Techniques aiming to increase generalization are collected under the term regularization. One possible approach is to reduce the model size, for example by removing layers or single neurons. However, this reduction has to be tuned carefully to not result in an underfitting neural network. Another strategy is to stop the training phase when overfitting is detected, i.e. immediately when the validation error starts to rise. This method is referred to as early stopping.

Ensemble learning can be used as well. Ensembles are a committee of multiple models which are possibly over- or underfitting. The individual predictions are then used to form a combined decision, for example by voting. This way weaknesses of an individual model can be compensated by other models. However, this approach gets unfeasible for large neural networks [SRI14]. A technique inspired by ensemble learning is called dropout. [SRI14] proposes to randomly deactivate some neurons and their corresponding weights during each epoch of the training process. In each epoch of the training, a different combination of neurons is deactivated. Dropout is only used during the training process and deactivated afterwards. According to [SRI14], the trained network can be seen as a combination of multiple thinned out networks and can therefore be understood as an alteration of ensemble learning.

In addition to presented techniques, the model's capacity can be controlled by punishing large weights within the error function. As mentioned in this section, capacity rises with increasing weights. If the weights are present in the error function, the model learns to keep weights, and thus the capacity, as small as possible if they do not lead to a considerable performance gain. A common approach to add the weights into the loss function are the L1-norm or the L2 norm, scaled by factor α. Adding the L1-norm is known as Lasso regularization, adding the L2-norm as ridge regularization respectively [POL19]. Ridge and Lasso regularization for the example of cross-entropy are shown in the following formulas:

$$Lasso: \quad L_{crossentropy}(y,\hat{y},\mathbf{w}) = \frac{1}{n}\sum_{i}^{n} y_i \cdot \log \hat{y}_i + \alpha \cdot \sum_{i} |w_i| \qquad \text{Eq. 2-20}$$

$$Ridge: \quad L_{crossentropy}(y,\hat{y},\mathbf{w}) = \frac{1}{n}\sum_{i}^{n} y_i \cdot \log \hat{y}_i + \alpha \cdot \sqrt{\sum_{i} w_i^2} \qquad \text{Eq. 2-21}$$

Despite all regularization techniques, the training data set still plays a key role in model overfitting. A neural network is only capable to generalize if the data set used for training is reflecting the data of its use-case properly. A larger amount of training samples often leads to a better representation of the use-case. This can be reasoned with the law of large numbers [EVA04]. [GOO16] describes training the model on a larger amount of data as the best way to prevent overfitting.

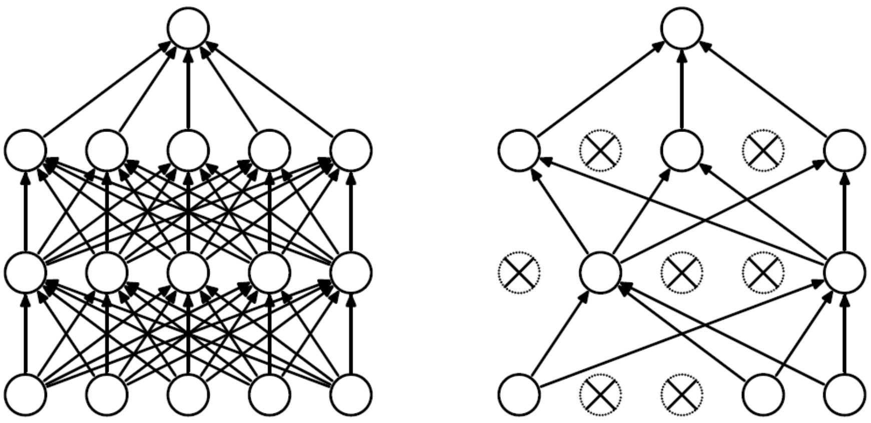


Fig. 2-9: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped. Taken from [SRI14].

2.7 Semantic Segmentation

Measuring the distance of the vehicle to surrounding objects is not sufficient for many automated driving tasks. To plan a vehicle's trajectory, it is important to know which kind of objects are being measured. As described in chapter 1, a tree on the roadside may require a different behavior than a pedestrian about to walk across the road. Therefore, the measurements have to be classified.

Semantic segmentation describes the point-wise classification of a data sample. It has been mainly applied to images so far, but finds more and more applications on pointclouds.

Due to the importance of scene understanding, semantic segmentation has been studied extensively. It has been tackled both with "traditional" computer vision methods as well as machine learning techniques. In many fields of computer vision, neural networks have proven to outperform all "traditional" approaches for many public data sets [THO16; GAR17]. Therefore, the following section focuses on presenting state-of-the-art research on semantic segmentation using neural networks. In the beginning of semantic segmentation research, the field of application was focused on images. Later, it was expanded to 3D-images and pointclouds, benefiting from the experinces made on images. Therefore, the following section reviews semantic segmentation on both images and pointclouds. A more detailed review of state-of-the-art research in semantic segmentation using deep learning can be found in [GAR17].

The neural network which paved the way for deep learning in semantic segmentation is called AlexNet [KRI12]. It demonstrates the capabilities of neural networks by winning the ILSVRC image classification challenge in 2012 [RUS15] with an accuracy of 84.6%. The best non-deep learning competitor only achieves 73.8%. [KRI12] attributes the success to the use of a high number of layers in the neural network. An additional major contribution made by the authors is the outsourcing of the training to GPUs, which makes the computationally expensive deep learning feasible.

One year after AlexNet, [SIM14] introduced VGG-16 which consists of 16 convolutional layers instead of AlexNet's five. VGG-16 is still a popular model today achieving an accuracy of 92.7% on the ILSVRC image classification challenge. GoogLeNet won the ILSVRC challenge in 2014 with an accuracy of 93.3% by introducing inception modules [SZE15]. The key idea is to not only combine multiple layers sequentially, but also in parallel.

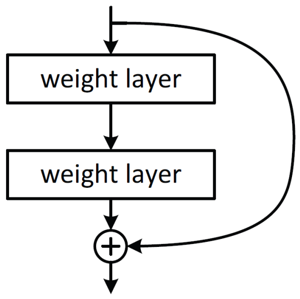


Fig. 2-10: A residual block as used in ResNet. A skip connection adds the block's input to its output. Taken from [HE16].

The idea of non-purely sequential network layers is also picked up by Microsoft's ResNet which uses so called residual blocks. Residual blocks are a combination of a few convolutional layers using a skip connection. This skip connection sums the block's input to its output, as depicted in figure 2-10. If all weights are initialized to zero, the input is simply passed to the output. The residual block only needs to learn features to add on top of the previous layer in order to reduce the model loss, i.e. they only learn the residual. The idea is that learning the residual is easier than learning the full input-output mapping [HE16]. Resnet consits of 152-layers. With today's wide availability of affordable GPUs, even networks as deep as ResNet can be trained in a reasonable amount of time. [POH17] uses a ResNet based neural network architecture to perform semantic segmentation of road scenes for automated driving. The network is easily trainable and reaches high performances on the City Scapes data set. This data set is one of the most popular and widely used image data sets for semantic segmentation in automated driving [COR16]. The network is named resnet-like due to its usage of residual blocks.

However, both training duration and memory demand of trained models are growing rapidly with increasing number of layers. In order to reduce the model size, [IAN16] propposes SqueezeNet, a network that reaches the same performance as AlexNet while reducing the number of parameters to train by a factor of 50, resulting in a model size of less than 0.5 MB. Too large models can lead to problems when embedding them in mobile real-time systems, as needed in automated driving for example. The size reduction is enabled by so called Fire modules as seen in figure 2-11. Fire modules consist of two layers: A squeeze and an expand layer. The squeeze layer consists of convolution filters with a kernel size of 1x1 while the expand layer is a combination of 1x1 and 3x3 convolution filters. The idea behind this is to reduce the kernel size as much as possible without loosing performance. For example, a 3x3 filter has 9 times as much trainable weights as a 1x1 filter. In addition to that, the squeeze layer in front of the expand layer can be used to reduce the number of input channels for the larger expand filter. The smaller kernel size can be balanced by a slightly higher layer number. Additionally, SqueezeNet makes use of skip connections inspired by ResNets. [IAN16]

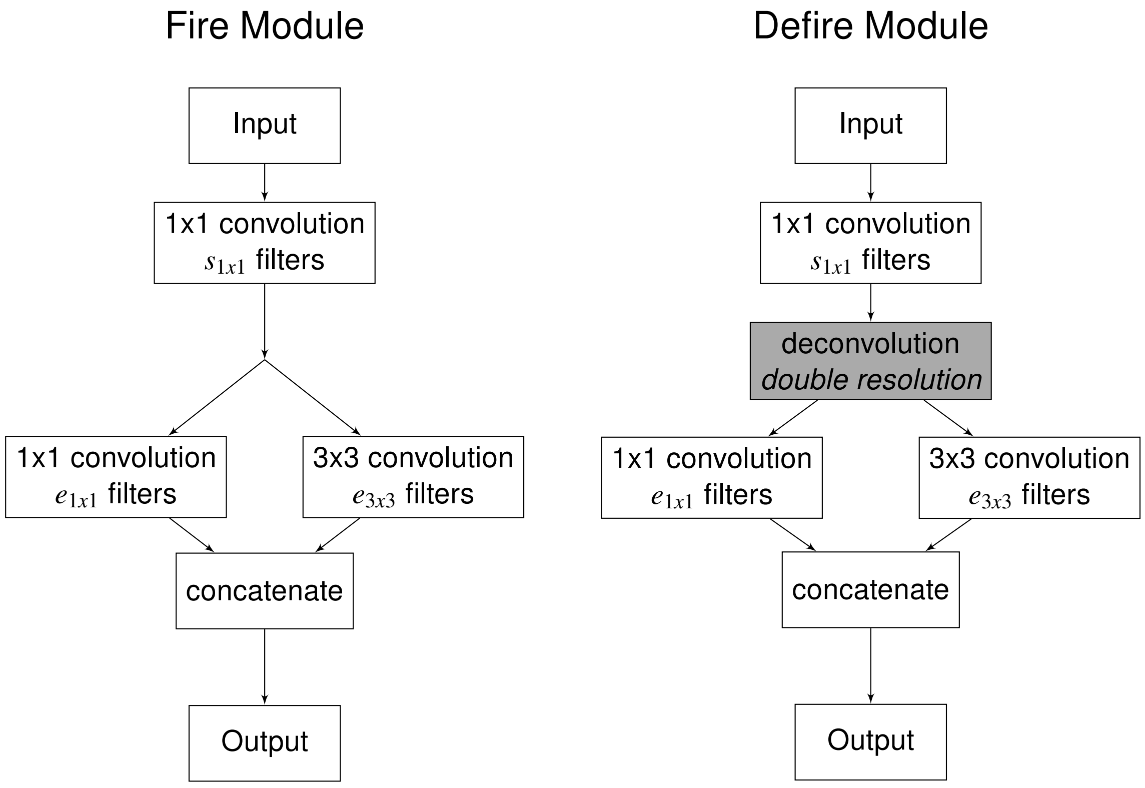


Fig. 2-11: Fire module introduced by SqueezeNet. It consists of a squeeze and an expand layer. The squeeze layer consists of convolution filters with a kernel size of 1x1 while the expand layer is a combination of 1x1 and 3x3 convolution filters. A Defire module has an additional deconvolution layer between squeeze and expansion layers. This deconvolution upsamples the input to have four times the original resolution. The number of filters used is denoted as s_{1x1} for the squeeze layer. For the 1x1 and 3x3 filters in the expansion layers, the number of filters is described by e_{1x1} and e_{3x3} respectively.

After the success of applying deep neural networks to semantic segmentation of images, there have been many attempts to apply the same techniques to 3D-pointclouds. Some problems have to be considered when transferring semantic segmentation from the image to the pointcloud domain. First, pixels representing an image are sorted in an array like structure and therefore have a predefined structure. Pointclouds however are just a set of points without specific ordering. In addition to that, pointclouds are typically sparse and of heterogeneous density. As mentioned before, CNNs demand a matrix-like input.

In order to still use the popular convolutional neural networks, most researchers try to transform pointclouds to a matrix-like shape [POH17]. Classical convolutional layers can be used if the additional third dimension is treated as a channel of an image, Instead of red, green and yellow in RGB images, there would be height intervals for example. There are also specific volumetric convolutional layers [JI13] which show a better ability to process the spacial information but are computationally more expensive [POH17]. A popular choice of network architecture are encoder-decoder structures [MA18; ZHA18; JAR18] and architectures developed originally for images [BEL18]. Encoder-decoder structures are a combination of convolutional, pooling and upsampling layers. [WU17] took inspiration from SqueezeNet and developed a network for semantic segmentation of pointclouds using Fire Modules. Due to its origin, the network is called SqueezeSeg.

Pointnet bypasses the pointcloud transformation by taking the raw pointcloud as direct input. The developers of Pointnet take advantage of the symmetry of the maxpool operation which simply takes a maximum characteristic of a subset of points. According to [CHA17], the most relevant point of a subset can be selected and new features for this point can be computed to use in the next maxpool operation.

2.8 Pointcloud Transformations for Convolutional Neural Networks

As mentioned in the previous section, pointclouds have to be transformed into image-like shapes, i.e. matrices, if convolutional neural networks are used. When choosing a suitable transformation, multiple aspects have to be considered. The transformation should lead to preferably dense representation of the pointcloud. The resulting matrix should have dimensions as small as possible while preventing the loss of too much original information. The computation time for the transformation should not be too high either if the prediction is used in time-critical applications as automated driving for example.

By dividing the pointcloud's Cartesian coordinates into cuboid voxels, the pointcloud can be represented as a 3D grid map. With a voxel size of $l_x \times l_y \times l_z$ and maximum considered distances d_{max_x}, d_{max_y} and d_{max_z}, the grid map's dimensions are $\frac{d_{max_x}}{l_x} \times \frac{d_{max_y}}{l_y} \times \frac{d_{max_z}}{l_z}$. Each voxel may contain one ore more characteristics computed from the points belonging to that voxel. For example, the total number of points or the maximum, the average or the accumulated intensity of LiDAR points are possible characteristics. If RGB-images are fused with the pointclouds, color values can also be assigned to each voxel. The advantage of this transformation is that visualizing the resulting grid map gives a very simple and concrete depiction of the underlying pointclouds. However, this simple representation typically leads to very large and sparse matrices. Many different transformations and characteristics exist for different applications. The two most common and recent representations are presented in the following.

[BEL18] introduces a bird's eye view representation. The x and y coordinates are computed as mentioned above while the third dimension already encodes the cells' characteristics. Three characteristics are used: Maximum height, mean intensity and the number of points divided by the maximum possible number of points. The normalization of the third characteristic makes the representation independent of pointclouds of different densities, for example arising from different LiDAR sensors. By sparing the voxelization of the z-dimension, this representation becomes denser. However, the representation loses a lot of information of the point distribution in the z-Axis.

Instead of using a Cartesian representation, [WU17] used spherical coordinates. The first two dimensions of the resulting matrix are discretized azimuth and zenith angles. Again, the third dimension holds characteristics, in this case x, y and z coordinates, intensity and range $r = \sqrt{x^2+y^2+z^2}$. As mentioned before, LiDAR pointclouds often already contain azimuth and zenith angle. If they are not available, they have to be computed as follows:

$$\begin{pmatrix} \theta \\ \varphi \end{pmatrix} = \begin{pmatrix} \arcsin \frac{z}{\sqrt{x^2+y^2+z^2}} \\ \arcsin \frac{y}{\sqrt{x^2+y^2}} \end{pmatrix} \qquad \text{Eq. 2-22}$$

[LI16] uses a similar transformation but only considers the height z and distance $d = \sqrt{x^2+y^2}$ as characteristics.

2.9 Labeling Data Sets

Point-level classification, as done in semantic segmentation, requires point-wise labeling of the training data set. The annotation is typically performed by a human annotator. Therefore, it is a task which is both time-consuming and prone to errors. Many researchers try to reduce the cost of the labeling process by developing interactive annotation tools [YUE18]. These tools aim to reduce the amount of time spent for each labeled input as well as increasing the accuracy of the point-level annotations.

An example for an interactive labeling tool used in the field of automated driving is the one used to annotate the City Scapes data set [COR16] mentioned in section 2.7. The tool enables a segmentation of the image by defining polygon vertices. It provides very basic function like zooming and merging of polygons in order to support a human annotator. Still, tools providing only basic functionalities for the labeling process are predominantly used. However, there are efforts done to find more sophisticated methods to reduce labeling cost.

One approach is progressive refinement proposed by [KOP11]. The concept of progressive refinement is to gradually divide the points into groups which can be labeled at once. The amount of instances to label can therefore be reduced and time is saved. [VEI14] advances this approach and introduces a multi-touch device to facilitate the annotation process.

[SHA12] as well as [WON15] suggest using automated proposals. Instead of labeling the complete input, the annotator only has to correct the proposed labeling. [BOY14] enhance this approach by introducing group labeling. An algorithm selects similar objects to annotate at once.

2.10 Data Augmentation

Data augmentation is a popular technique to increase the size of a training data set without the need to obtain and label further samples. The key idea of most augmentation techniques is to apply label-preserving transformations to a sample. The transformed element is then treated as a new sample and added to the data set. [GOO16] describes the development of suitable augmentation techniques as a key process in the development of new neural networks. Augmentation is mainly used on image data sets as efficient image-processing techniques already exist to apply the necessary transformations. However, most key ideas can also be applied to pointclouds. For this reason, the state of the art in image and pointcloud augmentation is presented in this chapter.

First image augmentation techniques are introduced by [SIM03]. Images are distorted to create additional training samples for the popular MNIST digit recognition data set [LI 12]. A convolutional neural network trained on the distorted images outperforms other algorithms for number classification on the MNIST data set available at that time. The effectiveness of creating further samples using augmentation implies that the MNIST database is too small for most algorithms to infer generalization properly [SIM03]. In addition to that, the paper is able to show that neural networks are outperforming other machine learning techniques in the area of pattern recognition and contributes to today's dominance of neural networks in this field.

[KRI12] adds flipping of the images as well as modifications of the individual RGB-values and applies them to the ImageNet data set. ImageNet consists of over 15 million images [DEN09]. Even on this large data set, performance is improved by using data augmentation to prevent overfitting of the trained neural network.

[SIN18] presents a complementary image augmentation technique. Random patches are added to images in order to hide random areas. The intention behind this technique is to prevent a neural network from focusing on single characteristics of an image, for example the face of a person. Hiding the face in some samples forces the neural network to consider other parts of a human as detection characteristics.

Data set augmentation is mainly used on image data sets as efficient image-processing techniques already exist. Multiple augmentation libraries have been developed, for example augmentor by [BLO17] and albumentations by [BUS18]. These libraries enable an easy to use and quick augmentation that is even possible online during the training process. However, not all available techniques are suitable for every data set. Finding a beneficial augmentation strategy requires experience, a good knowledge of the data set and sometimes trial and error. Applying rotations onto MNIST for example result in sixes and nines being indistinguishable.

Smart Augmentation is introduced by [LEM17]. The key idea of Smart Augmentation is to learn suitable augmentation techniques rather than manually engineer. A network is trained to learn which augmentation techniques lead to the highest performance gain. To keep the possible combinations of augmentation techniques feasible, a number of techniques to choose from is predefined.

As mentioned in the beginning of the section, most key ideas can also be applied to pointclouds. They can be translated, rotated, flipped or scaled in order to augment them. Some pointclouds contain further information, e.g. the intensity in LiDAR pointclouds. These characteristics can be augmented individually. This is comparable to the described rgb-value modification in images. Combinations of these techniques are often combined as demonstrated by [LI16; BEL18; ZHO18].

[BEL18] used those transformations to tackle a problem in the data of the widely used KITTI data set [GEI13]. For its LiDAR pointclouds, KITTI only provides annotations in the camera's field of view 90° in front of the measurement vehicle. By rotating the annotated points by 90°, 180° and 270°, he was able to train a model for 360° environment perception.

A recent approach in pointcloud augmentation consits of the modification of only those points belonging to an object. [ZHO18] uses this technique to slightly change the exact position of the objects in the scene. He used the LiDAR pointclouds and their 3D bounding boxes as data set. For each bounding box individually, he identified the points lying within this bounding box. Then, he assigned a random rotation and translation to each of those identified points. The rotational and translational parts were thereby sampled from a Gaussian distribution.

The previously described techniques are all based on the traditional idea of data augmentation: Performing label preserving transformations on existing data samples. A different, more sophisticated approach is fusing existing samples to create a new, semi-artificial one. [CHA02] introduces Synthetic Minority Over-Sampling Technique (SMOTE) where samples can be fused by simply computing their average. The idea of creating semi-artifical samples can be applied to images as well. In pixel-wise labeled images, pixels belonging to the same instance can be easily extracted. Those extracted pixels can then be inserted into another labeled images, preferably with a low amount of dynamic objects [LAM18; GRO18]. For example, pixels belonging to pedestrians can be extracted from multiple images and be inserted into an image of an empty road. This way, a scene crowded with pedestrians can be simulated. To our knowledge, this approach has not been transferred to pointclouds yet.

2.11 Clustering

After performing semantic segmentation on pointclouds, each point has an assigned class. However, in many applications it is necessary to divide the points of the same class into objects. For example, for an automated vehicle detecting 1000 points belonging to the class pedestrian, it is crucial to know how many pedestrians these points represent.

Clustering describes the grouping of data into subsets of somehow similar points. The measure of similarity is thereby not fixed and can differ depending on the type and distribution of the data. For LiDAR pointclouds, this measure could be the spatial distance of points, for example. The subsets created are called clusters. Clustering is solely based on the features of a data set, labeling is not needed. Thus, clustering belongs in the category of unsupervised learning. A visualization of a simple example is found in figure 2-12. It is a very broad area of study and many types of clustering algorithms exist. In this section, focus is set on the two algorithms most widely used: K-means and mean-shift.

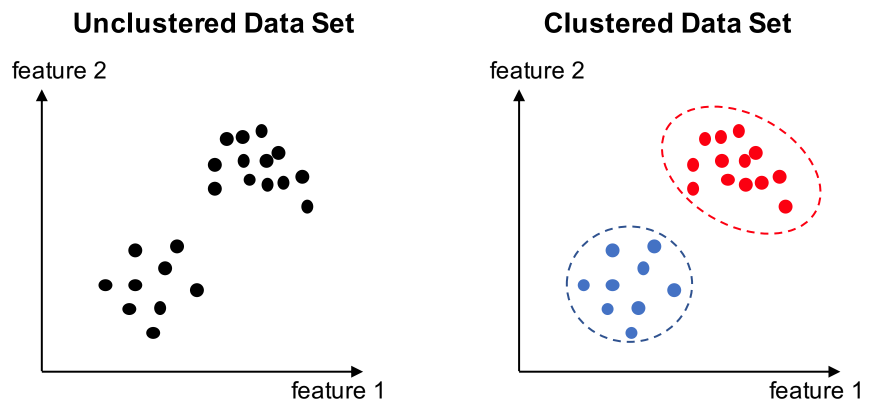


Fig. 2-12: Visualization of clustering. For example, this simple data set with two features can be divided into two clusters.

K-means clustering is one of the oldest and simplest clustering algorithm. The number of cluster K, into which the data set is divided has to be defined prior starting the algorithm. K-means clustering then computes K cluster centroids. Each data point is assigned to its nearest centroid, thus creating clusters [MAC67]. To cluster spatial data, like pointclouds for example, euclidean distance is typically chosen to detect the nearest centroid. However, other distance measures are possible.

In order to compute the cluster centroids, random centroids are initialized and then iteratively shifted. First, the points of the data set are assigned to their nearest centroid. Then, a new centroid of each cluster is computed by computing the mean value of all points belonging to a cluster. Shifting the centroid leads to another assignment of points to centroids. Shifting the centroids by computing the mean and reassigning points to cluster is repeated until the cluster assignment does not change anymore. Optionally, a criterion for abortion can be defined.

While K-means clustering is a powerful and computationally cheap algorithm to perform clustering, the number of clusters has to be predefined. However, it is not always easy or possible to define the number of clusters a priori. Mean-shift clustering is able to perform clustering without predefining the exact number of clusters [COM02]: A (possibly high) number of randomly positioned centroids is initialed. Instead of now assigning all points to their nearest centroid, a region of predefined size around each centroid is considered. For each centroid, the mean of all points lying in this region is computed an the centroid moved towards the direction of this mean, scaled by a factor. This means, that some points are used multiple times for the mean computation and some are not at all. The centroids converge to regions of locally high density.

As a high number of initial centroids exists, some might converge to a similar position. If the centroids overlapped after they converged to a position, only the one with the higher number of points inside its region is mantained. Afterwards, each point is assigned to its nearest centroid [COM02]. As mentioned, mean shift does not demand to define a number of cluster a priori. However, this comes at the price of higher computation cost, as a high number of centroids is used.

2.12 Metrics for Performance Evaluation

In order to evaluate the performance of a neural network, metrics have to be defined. The error function used during training might not always be suitable for evaluation. Cross-entropy for example is a popular and effective loss function to train a neural network. However, it does not lead to a well comparable scale and is a very abstract evaluation metrics. Metrics with fixed scales (for example between zero and one) and easily interpretable might be more suitable for performance evaluation. Additionally, it is often beneficial to use multiple metrics.

Considering semantic segmentation, typical metrics used are precision, recall and intersection-over-union. They are all based on an error matrix which is called confusion-matrix and depicted in table 2-1. It consists of four categories describing the comparison of prediction and ground truth:

true positive (tp): The network correctly correctly detects the class to be evaluated.

false positive (fp): The network mistakenly detects the class to be evaluated.

false negative (fn): The network mistakenly rejects the class to be evaluated.

true negative (tn): The network correctly rejects the class to be evaluated.

If more than two classes exist for prediction, the confusion matrix is usually computed for each class individually. The per-class results can then be averaged.

	ground truth: yes	ground truth: no
prediction: yes	true positive	false positive
prediction: no	false negative	true negative

Tab. 2-1: Confusion matrix for a binary classifier.

The individual values in the confusion matrix can be used in order to compute metrics that evaluate the performance of the examined neural network. Typically, multiple metrics are taken into consideration when assessing performance. Using only one single index is often not appropriate to judge predictions based on unbalanced data sets. In this work, precision, recall and intersection over union will be used for evaluation.

Precision: The precision of a neural network is the relation of true positives to the total number of detected positives. It therefore describes how reliable the model's predictions are. Its value can therefore be calculated as follows:

$$precision = \frac{tp}{tp + fp} \qquad \text{Eq. 2-23}$$

However, precision does not take false negatives into account. Assuming an automated vehicle trying to pass a crosswalk, not considering false negatives regarding pedestrians crossing the street can arise problems. It is critical to detect every pedestrian in front of the vehicle to prevent accidents, thus there should be no false negative detection. It is no critical problem that creating such a sensitive detector results in some more false positives, as falsely detecting a pedestrian only leads to waiting longer at the crosswalk.

Recall: In order to compensate the weakness of precision, recall can be taken into account. The recall, also called true positive rate, of a neural-network describes the relation of true positives to the real number of positives and can be computed by the following formula:

$$recall = \frac{tp}{tp + fn} \qquad \text{Eq. 2-24}$$

It therefore describes how reliably a classifier detects *positive* ground truths. In contrast to precision, recall does not consider false positive detections. In the scenario mentioned above, not considering false positives at all could lead to an automated vehicles falsely detecting pedestrians everywhere, making driving impossible. This is the reason why both precision and recall need to be used for evaluation.

Intersection over Union: A metric often used in semantic segmentation is the intersection-over-union (iou). This metric evaluates how much prediction and ground truth overlap in relation to their size. It is also called Jaccard index and is computed as follows:

$$iou = \frac{|prediction \cap groundtruth|}{|prediction \cup groundtruth|} = \frac{tp}{tp + fp + fn} \qquad \text{Eq. 2-25}$$

It takes both false positives and false negatives into account and is well suited to evaluate semantic segmentation for automated driving tasks. Intersection over union is used as the main evaluation metric in this work as it is presumably the most common metric to evaluate semantic segmentation. Precision and recall are studied additionally as they might point out where weaknesses of the model originate from.

3 Motivation and Research Question

As described in chapter 1 and 2, this work aims to develop and evaluate a concept for data augmentation techniques applicable to LiDAR-pointclouds. The motivation behind this is the high difficulty to obtain a large amount of high quality, point-level annotated pointclouds needed to train a neural network. This thesis aims to use the developed concept for augmentation to create large data sets of annotated pointclouds without the need of manual labeling.

Sections 2.5 and 2.6 describe how neural networks are trained and why many high quality training samples are crucial to train a suitable model. Data augmentation creates additional samples without the need for further human annotation. When developing a concept for pointcloud operation, inspiration can be taken both from state of the art pointcloud augmentation and from image augmentation which is already well studied. The concept is developed and evaluated in this work, aiming at answering following research question:

Which augmentation techniques can be applied to LiDAR-pointclouds in order to increase the performance of a neural network that performs semantic segmentation on these pointclouds?

The augmentation techniques used in this work can be divided into two groups. There are augmentation techniques in the traditional sense, i.e. performing label preserving transformation on the training samples as presented in chapter 2. Additionally, the creation of semi-artificial training data can be understood as data augmentation as well. As the creation of semi-artificial data differs strongly from the traditional sense of data augmentation, this work differentiates between the two techniques, though they both belong to the field of data augmentation. Regarding possible augmentation techniques, studying following question can help answering the research question:

How does the performance of a CNN trained on semi-artificial samples compare to state-of-the-art performances of CNNs trained on manually labeled pointclouds from recorded traffic scences?

Inspired by [LAM18] and [YUE18], whose work is presented in chapter 2, automatically labeled pointclouds of objects can be merged with recordings free of objects that shall be classified. Automatically labeling objects is possible in controlled environments where only the objects of interest exist. It is then possible to extract these objects based on the height of the points representing the objects and the proximity of individual points. This technique might have the potential to drastically reduce or even revoke the need for human annotated pointclouds.

During the creation of the semi-artificial data set, the number of objects and their insertion locations can be influenced by choosing different probability distributions. This can be done for each class individually. Multiple data sets based on different distributions can be created and evaluated with respect to the trained network's performance.

Instead of using objects extracted from measurements of a controlled environment, only labeled objects of an already existing data set can be used as well. In this case, the objects are inserted into different scenes of the same data set.

In which way is the the performance of a CNN influenced by traditional augmentation techniques applied to its training pointclouds?

In addition to merging pointclouds of different recordings, traditional augmentation techniques can be either applied to the fused semi-artificial samples, to the objects before insertion or to the empty scenes before insertion. Examples are simple rotations of the scenes, adding noise to the measurement or simulating occlusion introduced by potential objects that are located between the sensor and another object.

To which extend do the developed augmentation techniques lead to a supplement or even substitution of manually annotated pointclouds from recorded traffic scenes?

A direct comparison of semi-artificial samples and recordings of road traffic can be done. To study how realistic the semi-artificial samples can be produced, a real road traffic scene can be reconstructed using semi artificial samples. A possible way to do so is to remove all points not belonging to the background in the real scene. Then, corresponding objects extracted from the measurements in the controlled environment can be inserted at the exact same position. This way, it is possible to create a semi-artificial scene in the exact same environment as a real scene, but with inserted objects.

Additionally, the traditional augmentation techniques can be applied to manual annotated data. Objects can be split from the background in manually annotated scenes. This leads to a set of empty scenes and another with objects for insertion. Semi-Artificial samples can be created. Instead of merging recordings of empty road traffic scenes with objects recorded in a controlled environment, the sources for the creation originate from the same measurement.

4 Concept

This chapter explains the concept developed for this thesis. The design process loosely follows the V-model for systems engineering [OSB05]. The development process is divided into two parts, the first being the development of the concept itself and the latter one being the evaluation and testing. This chapter focuses on the left part of the V-model, i.e. the development of a concept for the augmentation of LiDAR-pointclouds.

First, the concept of operations is presented in section 4.1. Then the requirements to each of the operations are presented in section 4.2. Then, the design concept is described in section 4.3, again individually for each operations. The evaluation of the concept can be found in chapters 5 and 6.

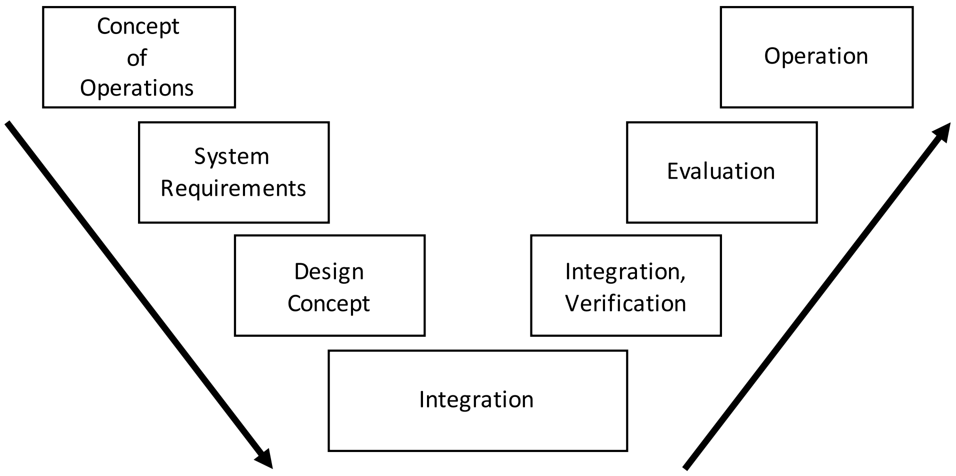


Fig. 4-1: V-Model for systems engineering. The left part describes the requirements to the system and the development concept. The right part deals with the verification, evaluation, operation and maintenance. Adapted from [OSB05].

4.1 Concept of Operations

Though the scope of this thesis clearly focuses on the development and evaluation of augmentation techniques, a whole system for the semantic segmentation of pointclouds needs to developed. This system can be divided into five operations: Data recording, augmentation, pointcloud transformation, model training and evaluation. Though the scope of this work clearly focuses on augmentation and evaluation, all modules shown in figure 4-2 are crucial for this thesis. The measurements needed to create semi-artificial samples have specific requirements. No data fulfilling this requirements is yet available at the Institute for Automotive Engineering (IKA). Additionally, no suitable network is yet implemented at the institute, which is why a suitable network architecture has to be found. As no network is implemented, a suitable pointcloud transformation is missing as well. Therefore, the concept developed for each of the five areas is presented in this chapter.

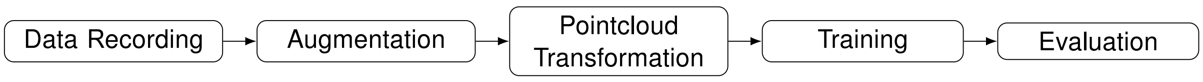


Fig. 4-2: The whole developed system as a block diagram. The concept can broadly be divided into five parts: Data recording, augmentation, pointcloud transformation, model training and evaluation. All parts are presented in section 4.3

4.2 System Requirements

Before starting concept development, requirements for the system developed have to be defined. For each of the operations identified in the concept of operations in figure 4-2.

4.2.1 Data Recording

The setup used record to record training data should be as similar as possible as the setup that records the pointclouds during inference, i.e. when the trained model is used in an automated vehicle. As mentioned in section 2.6, this requirement is crucial to receive a model with decent generalization.

In the context of this thesis, this means that the LiDAR sensor used for measuring should be mounted at the same position in the vehicle as later during inference. Additionally, measurements should be recorded in similar weather conditions as well as environment types.

In order to examine all points raised in chapter 3, three different types of measurements have to be recorded:

1. Measurements within a very simple environment, enabling automatic labeling.

This measurement is used to extract objects for augmentation. As stated in chapter 3, objects within a controlled environment are extracted and inserted into real road traffic scenes.The motivation behind creating this semi artificial samples is the reduction of labeling effort. Therefore, it has to be made sure that the controlled environment enables an automatic labeling and extraction of the relevant objects. Additionally, the extracted object should represent the whole object class properly. This means recording the objects in different poses, angles and distances. A pedestrian recorded running produces a different LiDAR pointcloud than walking or standing. Pointclouds differ if he is recorded from the front or side, wearing a backpack or not. Additionally, the intra-class variability should be considered during the measurements. Instances of each class have different sizes and dimensions. Therefore, multiple objects of the same class should be recorded.

Occlusion of the recorded objects should be avoided as well. Later in the augmentation process, it is possible to emulate occlusions by removing points. Recovering occluded areas of an existing object is much more challenging.

2. Measurements on public roads, free of objects we want to detect.

This measurement is used to insert the objects extracted from 1. Therefore, the main requirement is that the recordings are free from objects the neural network is supposed to detect. Since the developed semantic segmentation network is supposed to perceive urban environments, recordings in similar areas should be aspired. Within the scope of urban environments, data should be recorded of preferably many different environments, i.e. different streets and varying building density for example. This requirements is challenging as it is difficult to record urban areas without these kind of objects.

3. Measurements on open roads with objects that can be detected.

This measurement will be used for evaluation of the augmentation techniques. The requirements to the recorded environments is identical to 2. Additionally, the measurement should be performed without drawing too much attention from other road users. Vehicles noticeably equipped for automated driving, typically draw attention by passing people. This often leads to unnatural behavior of both vehicles and VRU. Referring to section 2.6, it is important to record natural traffic to receive a neural network that performs well in future driving applications.

4.2.2 Pointcloud Transformation

As mentioned in chapter 2, convolutional neural networks need matrix-like inputs. Therefore, the pointcloud representation of the LiDAR measurements have to be transformed into a matrix. When choosing a suitable transformation, multiple aspects have to be considered. The transformation should lead to preferably dense representation of the pointcloud. The resulting matrix should have dimensions as small as possible. However, a trade-off between representation size and information loss has to be found. The computation time for the transformation should not be neglected either if the prediction is used in time-critical applications as automated driving for example. An existing representation presented in chapter 2 can be used if suitable or a new one can be developed.

Additionally, its beneficial if the transformed pointcloud still enables augmentation. Applying augmentation techniques after transforming the pointcloud reduces computation time during the training process. Considering an augmentation technique that creates n additional samples out of an original one before the transformation results in a computation time of:

$$t_{tot,1} = n \cdot (t_{augment} + t_{transformation}) \qquad \text{Eq. 4-1}$$

If augmentation is done after the transformation, the transformation itself only has to be computed once. The computation time reduces to:

$$\begin{aligned} t_{tot,2} &= n \cdot t_{augment} + t_{transformation} && \text{Eq. 4-2} \\ &= t_{tot,1} - (n-1) \cdot t_{transformation} && \text{Eq. 4-3} \end{aligned}$$

However, modifying the pointcloud directly leads to a wider range of applicable methods. A trade-off between techniques before and after the transformation process has to be found.

4.2.3 Neural Network for Semantic Segmentation

A neural network capable of semantic segmentation on pointclouds needs to be found. The search for an architecture is focused on convolutional neural networks, as they are a popular choice for semantic segmentation of both images and pointclouds. Even though we are looking for a well performing network, the development time has to be considered when choosing a suitable architecture. Within the scope of the thesis, we are not seeking to optimize a network for semantic segmentation. The neural network has to be suitable for the evaluation of the developed data augmentation techniques. This means we have to make sure changes in the training data set can both increase and decrease the prediction performance. That is to say those influences are not restricted by the chosen network.

4.2.4 Augmentation Techniques

As mentioned in chapter 3, the augmentation techniques can be divided into traditional techniques and the creation of the semi artificial data set.

Traditional Augmentation Techniques

The augmentation techniques developed in this work should be applicable individually. This enables parameters of the individual techniques to be varied and examined separately.

Augmentation techniques can be performed both on the pointclouds as well as on their matrix representations. However, it is desirable to focus on the raw pointclouds. As described in section 2.8, there is a variety of inputs formats that can serve as input for convolutional neural networks. Different representations possibly demand different implementations of augmentation techniques, if implementation is possible at all. In contrast to that, techniques applied to the raw pointclouds can be applied independently of the following transformation or network topology. This enables a wider relevance of this work's results. Additionally, it is beneficial if the developed techniques are applicable onto other kinds of pointclouds with little to no effort. That applies to either other LiDAR-sensors or other types of pointclouds.

The application of augmentation techniques leads to an increasing prepossessing time during the training process. The gain should be kept as low as possible in order to preserve a feasible training time. This can be achieved by developing efficient implementations of the techniques or by finding suitable simplifications.

Creation of Semi-Artificial Samples

Before creating the semi-artificial samples, objects for insertion have to be extracted from the controlled environments. It is not sufficient to label the pointclouds in this environment. Furthermore, it is necessary to know which points belong to the same objects. All points of the same class have to be grouped into instances. The number of instances might not always be known, as the objects might for example leave and enter the sensed area during the recording process. This can happen accidentally or by a belated restriction of the measured range. Restricting the measured range is for example necessary to transform the pointcloud into the matrix representation used for training.

As the motivation behind the creation of semi-artificial samples is to supplement or substitute real pointcloud measurements, it should be aimed to produce semi-artificial pointclouds as realistically as possible. For example, objects should be inserted at reasonable positions. The most important criterion is that objects are not inserted into obstacles. Simply comparing two pointclouds is not sufficient to find out if they overlap as pointclouds do not indicate if the space between two points is occupied or not. For each pointcloud of an object, an area marked as *occupied* has to be defined. Then, it is possible to check if any point not belonging to the object lies within the marked area. However, computing the exact occupied area is not possible as the pointcloud does not contain this information. A suitable approximation has to be found.

An additional requirement to the semi-artificial sample is that the insertion process should be parameterizable with respect to the class of the inserted objects, their number per scene and their insertion positions.

4.2.5 Evaluation

Evaluation Data Set

The data augmentation techniques developed in this thesis aim to produce pointclouds that can be used to train a neural network for semantic segmentation. They can serve as both substitution or addition to non augmented pointclouds labeled by a human annotator. As described in section 4.2.1, a manually labeled, non augmented data set is needed to evaluate to which extend these semi-artificial pointclouds show this capability. Without a manually labeled data set, there is no baseline the developed concept can be compared to. This work aims to bypass the need for human annotations through the creation of semi-artificial samples. However, a human annotated data set is needed to for evaluation as described in section 4.2.1. As the annotation of pointclouds requires a lot of effort, a technique facilitating the annotation process is needed. Section 2.9 shows recent work in data annotation. A labeling technique should be found that enables the labeling of one pointcloud in just a few minutes.

Evaluation Metrics

In addition to the evaluation data, suitable metrics have to be defined to evaluate the network's performance after training with differently augmented data sets. Multiple metrics help to evaluate the performance from different perspectives. Additionally, metrics that are widely used for semantic segmentation enable better comparison and integration into other research projects.

4.3 Design Concept

This section describes the developed concept for each operation showed in figure 4-2. Thereby, the requirements mentioned in the previous section are taken into account.

4.3.1 Data Recording

Vehicle Setup

An easy integration of the developed system into different research projects at the IKA should be strived for. Therefore, the institute's research vehicle sensor setup is used without modifications on the hardware side for data recording. The vehicle is a Volkswagen Passat CC. The sensor used in this thesis is a Velodyne Puck VLP-16 LiDAR-sensor on top of the vehicle.

The LiDAR sensor used in this work is a Velodyne Puck VLP-16. It consists of 16 laser beams with equidistant elevation angles between -15° and 15°. A visualization of the resulting laser beams is depicted in figure 4-3. Detailed sensor specifications are listed the appendix in table A-1.

An new coordinate system is chosen to represent all spatial information in this thesis. The x-axis points in the vehicle's driving direction, the y-axis perpendicular to its left, parallel to the ground. The z-axis points in the normal direction to the ground, forming an orthogonal system as visualized in figure 4-4.The whole system moves with the vehicle. The origin is located vertically under the LiDAR sensor at ground level:

$$\mathcal{O} = (x = x_{sensor} \,|\, y = y_{sensor} \,|\, z = 0) \qquad \text{Eq. 4-4}$$

The measurement vehicle used in this thesis was already equipped with a measurement system to record the pointclouds of the LiDAR sensor. The measured points belonging to one 360° sweep of the LiDAR sensor were collected and sent out at once. Each point is characterized by 7 coordinates. The tuple of each point contains the x,y and z coordinate, the intensity of the reflected laser beam I, the azimuth φ, the measured distance r and the index of the laser which measured the point n_{laser}:

$$P = (x, \quad y, \quad z, \quad I, \quad \varphi, \quad r, \quad n_{laser}) \qquad \text{Eq. 4-5}$$

The Cartesian coordinates x, y and z are redundant. As explained in section 2.1, they can be computed using the spherical coordinates given by φ, r and the elevation angle Θ_{laser} given by n_{index}.

A timestamp was associated with each pointcloud.

Measurements in a controlled environment

As stated in 4.2.1, measurements within a very simple environment are needed. The idea is to record objects in an environment so simple that automatic labeling is possible. The test track of the IKA offers an asphalted plane. If only objects of the same class are recorded, labeling gets straightforward as all points within the pointcloud either belong to the object or the ground. Therefore, all points above a height-threshold in this recordings are automatically assigned to the corresponding class. For a better understanding of the measurement setup, a top view image taken during one recording session is shown in figure 4-5.

Five different pedestrians and two bikers were recorded for extraction. They were instructed to change their poses during the measurement process. For the pedestrians, this included for example to change from walking to running or to vary their steps sizes. The bikers changed

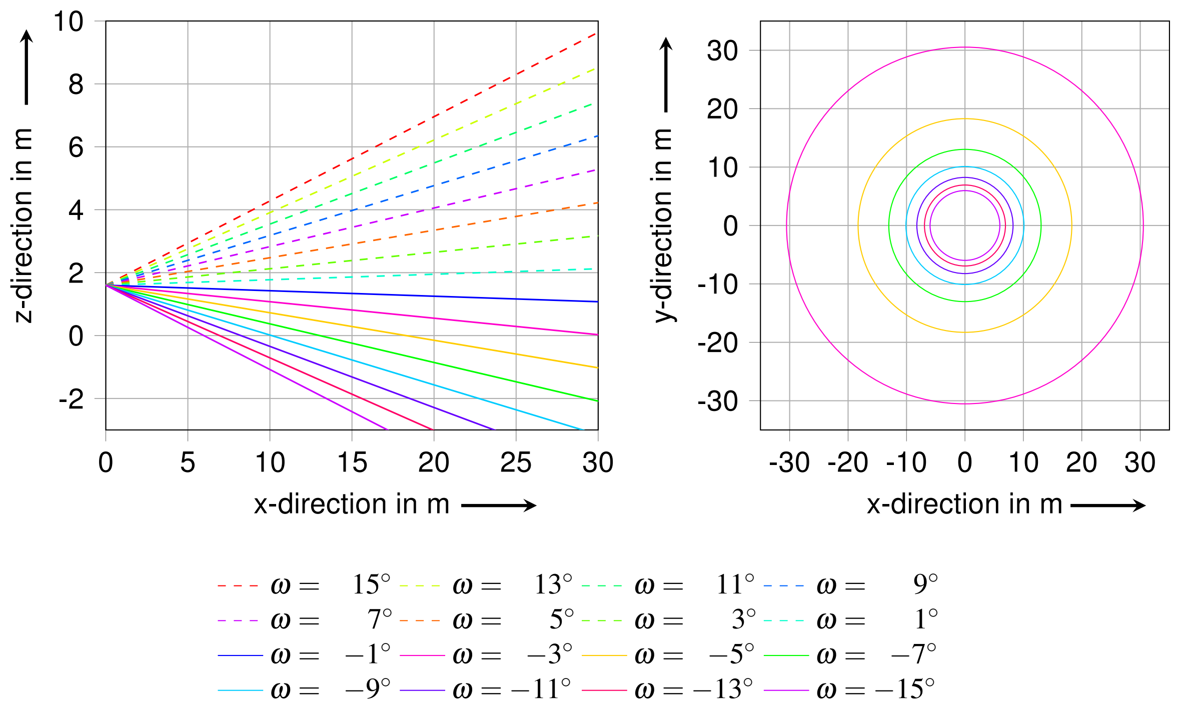
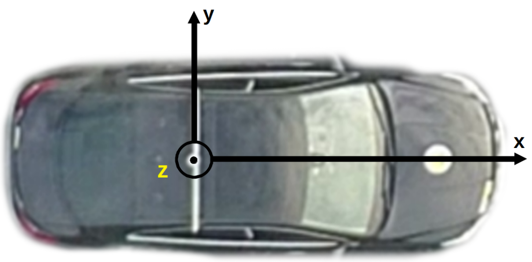


Fig. 4-3: Laser beams of a Velodyne Puck VLP-16 mounted at 1.6m height.
Left: Side view of the (rotating) laser beams. Beams with a positive elevation angle are marked dashed. Right: Top View of the locations where laser beams hit the ground. The ground in this plot is assumed to be a flat plane at z=0. Only beams with negative elevation angles ever hit this ground. As the beams are rotating around their vertical axis, they create circles at z=0. Modified from [BEY18].

a) Side view of the test vehicle, a VW Passat CC. The coordinate axis directions are indicated.



b) Top view of the measurement vehicle with position of the coordinate system's origin. The origin is vertically under the LiDAR sensor at $(x = x_{sensor}, y = y_{sensor}, z = 0)$

Fig. 4-4: Measurement vehicle used in this thesis. A VW Passat CC with a LiDAR sensor mounted at 1.61 m height was used. The coordinate axes are visualized. i) side view, ii) top view.

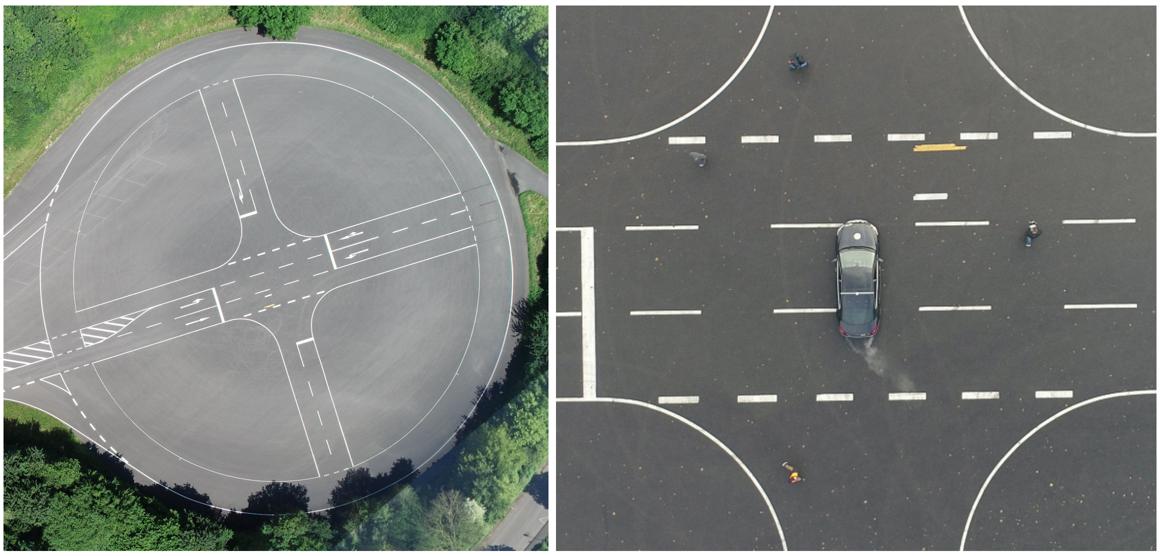


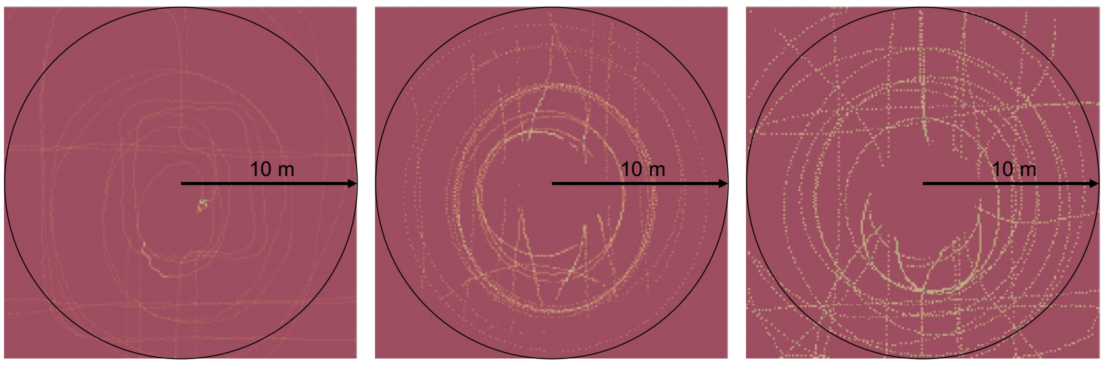
Fig. 4-5: Image of the measurement setup on the test track of the institute for automotive engineering. Four objects of the class *pedestrian* are visible. As no other objects exist around the test vehicle, all points recorded either belong to the ground or to a pedestrian.

their body posture on the bike. Additionally, two vehicles, a VW Passat B8 and a Mercedes E-Class, have been used during the measurements. This way, a larger variety of objects of the same class could be recorded. To keep the amount of measurements as low as possible, the pedestrian recordings have been done with multiple people at the same time. Those people had to make sure to not walk through the path between the LiDAR-sensor and another pedestrian, creating occlusions as demanded in 4.2.1. An overview of the objects recorded on the test track can be found in table 4-1.

measurement name	class name	objects recorded simultaneously	extracted objects	recorded pointclouds
pedestrian01	pedestrian	2	5177	7192
pedestrian02	pedestrian	2	2940	4869
pedestrian03	pedestrian	2	5681	4795
pedestrian04	pedestrian	4	3438	1160
pedestrian05	pedestrian	4	13578	4105
cyclist01	cyclist	1	2494	5720
cyclist02	cyclist	1	3503	6719
vehicle01	vehicle	1	2066	6546
vehicle02	vehicle	1	1034	4855
vehicle03	vehicle	1	3262	7802
vehicle05	vehicle	1	498	4146

Tab. 4-1: List of measurements performed on the test track.

In order to receive a feedback about the spatial positions of measured objects after each measurement, heat maps depicting the distribution of the objects' position on the test track are generated. This way, it is possible to asses the distribution of the positions of recorded objects directly after the measurement processes. Lesser covered locations can be focused during the following recording. Examples for heatmaps can be found in figure 4-6.



a) Pedestrian b) Cyclist c) Vehicle

Fig. 4-6: Examples for heatmaps regarding the measurements on the test track. a) pedestrian, b) cyclist c)vehicle.

Measurements free of objects of detectable classes

In order to be able to obtain data from public roads free of any objects belonging to classes the network is supposed to detect, recordings were made during night time. For the few scenes containing pedestrians, time stamps were noted during the measurement process. The area of recording spanned from Campus Melaten to Kaiserplatz in Aachen with a lot of smaller detours. A map indicating the measurement location is found in figure 4-7. By covering a large area, heterogeneous data could be recorded with different types of roads: Main roads, side paths as well as traffic-reduced areas. A list of the measurements recorded is found in table 4-2.

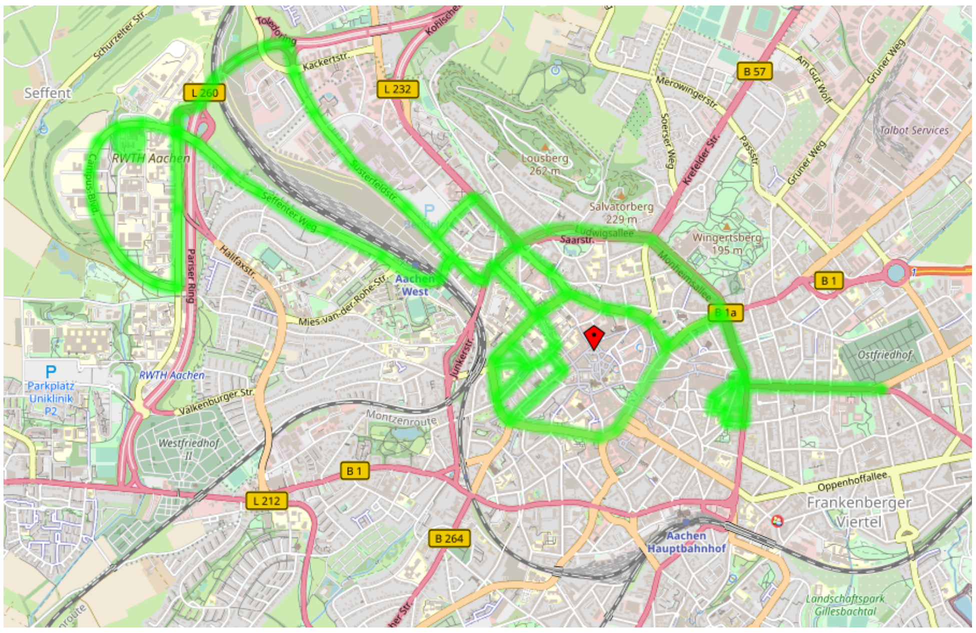


Fig. 4-7: Map of the city of Aachen, Germany. Measurements taken of public roads during the night are marked in green. Some roads were passed twice, if possible in different directions. Small side roads, main roads as well as traffic-reduced areas are included. Taken from Open Street Map [CON19].

measurement name	duration [min:sec]	pointclouds
boulevard01	01:08	1372
boulevard02	00:56	1119
superc	00:26	511
aachen	26:27	31737

Tab. 4-2: List of measurements during night time in Aachen.

Evaluation Data Set

The evaluation data was recorded at the intersection Kühlwetterstr./Süsterfeldstr. in Aachen, Germany. On one hand, it was chosen as it represents a suitable urban environment with urban buildings, a small square and main road. On the other hand, it was possible to obtain a permit to fly a drone during the measurements. This drone was used to support the pointcloud annotation process. This motivation and concept behind this drone assisted labeling is explained in section 4.3.5.

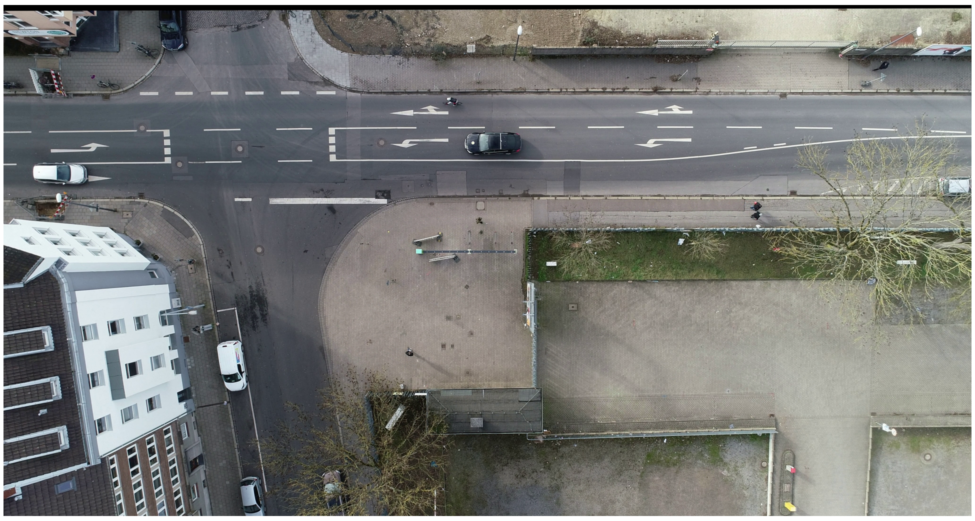


Fig. 4-8: Recording location of the evaluation data set. The measurements were taken at the intersection Kühlwetterstr./Süsterfeldstr. in Aachen, Germany. The image shows the measurement with the vehicle driving.

The measurement was obtained with two different types of measurement setups. First, the vehicle was standing on the roadside, close to the intersection. This way, pointclouds of pedestrians passing very close to the vehicle could be recorded. Additionally, recordings with a driving vehicle have been performed, as they represent a more realistic scenario. An image of the recorded environment is depicted in figure 4-8.

253 pointclouds have been annotated. For a better understanding of the evaluation data set, its size and the number of objects it contains are listed in table 4-3. Additionally, the locations of these objects are visualized in the appendix, depicted as heatmaps in figure A-1.

Table 4-3 shows that almost no bicyclists have been annotated for evaluation. Though it is possible to create semi-artificial samples containing bicyclists, they will not be considered for evaluation. A possible reason for the low number of bicyclists could be that the temperature during the measurement were relatively low.

class	**number of occurences**
pedestrian	215
bicyclist	4
vehicle	212

Tab. 4-3: Objects annotated in the evaluation data set. 253 pointclouds have been annotated.

4.3.2 Augmentation Techniques

In this work, augmentation techniques both on raw pointclouds as well as on the matrix representation are developed. As already shown in figure A-9, the traditional augmentation techniques are applied at two different moments: Before creating the semi-artificial samples, the objects which are inserted into empty scenes are augmented individually. After fusing objects and empty scenes, we do not augment the pointcloud further. Instead, the sample is augmented in its matrix representation during training. The idea behind augmenting the matrix representation is the reduction of computation time described in 4.2.2. The augmentations performed on the matrix-representations are performed fast enough to be applied during training time. By augmenting the sample directly before computing the gradient for weight modification, a sample can be augmented differently in each epoch. This possibly leads to a wider variance in the training data set and can therefore help to prevent overfitting.

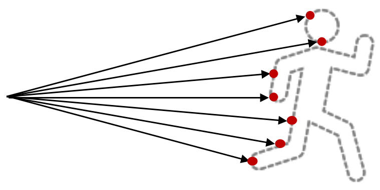
In theory, all augmentations can be performed during training time, including the creation of semi-artificial samples. This techniques bear a high potential to train networks showing better generalization. Each epoch, the combination of empty scenes, inserted objects and their augmentation could be alternated. A pipeline for full augmentation during training has been developed in this work. However, the computational cost is too high to be performed during training time. Selecting objects for insertion, augmenting them, fusing them with empty scenes and transforming the resulting sample into a matrix demands too high computation time and leads to unfeasible training duration. Therefore, this concept has been rejected. Instead, a data set of fixed size is created before starting the training. However, due to the high potential this techniques offers further research for run time optimization is recommended and will be picked up in chapter 7.

Augmenting Objects Before Insertion

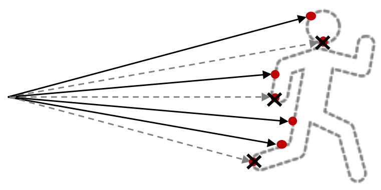
In order to augment the objects before inserting them into empty scenes, five techniques are presented: Rotation, removing random points, moving individual points horizontally, shifting the distance of the whole objects and simulating occlusions.

The first augmentation technique used is the rotation of the object around its horizontal axis. As the points' coordinates are given in spherical coordinates, only the azimuth position of each point has to be changed, while both the distance and laser indices stay unchanged. As mentioned in section 4.3.1, during the recording on IKA's test track, it was aimed to distribute the objects evenly during measurements. However, it is not possible to achieve a completely even distribution or another distribution could be desired. For example, section 2.10 describes how [BEL18] rotates the KITTI data set's pointclouds 90°, 180° and 270° to receive pointclouds all around the vehicle. The idea presented in this work is very similar, but many more rotational angles are possible and objects can be rotated individually. The heatmap in figure 4-10 shows how the measurements of a recording can be augmented by rotation to achieve very evenly distributed possible insertion positions. The figure can be compared to the heatmaps presented in figure 4-6

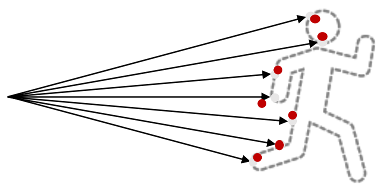
Another augmentation technique developed is the removal of random points. The inspiration behind this is that not all laser pulses that hit an obstacte are reflected back to the LiDAR sensor. This technique is simple to apply. A random percentage of points to remove is selected and



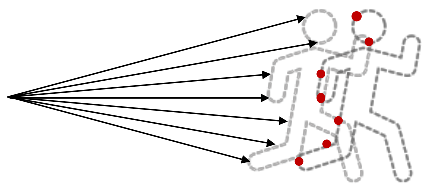
a) Original pointcloud.



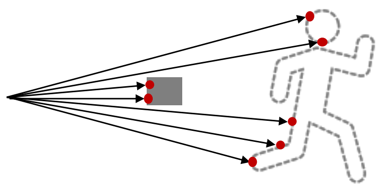
b) Removal of random points.



c) Moving points individually.



d) Shifting the distance of the whole pointcloud.



e) Simulating occlusion by an obstacle between sensor and object.

Fig. 4-9: Visualization of four developed techniques to augment the objects before inserting them into empty scenes. The arrows depict laser beams of a LiDAR sensor which are reflected by a pedestrian. The resulting LiDAR pointclouds is visualized with red dots. a) original measurement, b) removal of random points, c) shifting the points individually, d) shifting the distance of the whole pointcloud, e) simulating occlusion by an obstacle between sensor and object - Not depicted is the rotation around the vertical axis in the sensor's origin.

then uniformly random distributed points are selected to be removed. A percentage to remove instead of absolute value was selected, as objects close to the sensor produce far more points than objects far away. As already shown in figure 4-3, fewer laser layers are able to hit a target of constant size with increasing distance. The augmentation technique is depicted in figure 4-9b.

In real road traffic, a random removal of points might happen during heavy rain or snowfall due to occluded laser rays, as mentioned in section 2.1. However, the influence on weather conditions is out of the scope of this work.

Instead of removing indivudal points, spatial noise can be introduced by moving individual points randomly. By this, inaccuracies and noise of the LiDAR sensor can be simulated. The

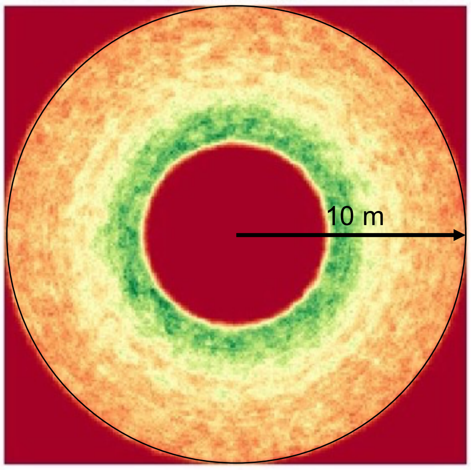


Fig. 4-10: Heatmap showing possible insertion locations after augmenting objects by rotations. The figure can be compared to figure 4-6, which shows heatmaps for object location before the augmentation by rotation.

points are only moved horizontally, as the height of a point is only determined by its laser index n_{laser}. For high distances, moving a point by only a single layer, large shifts in height are possible. These distances are considered too be to high and therefore a vertical movement is left out. For example, figure 4-3 shows that the height difference between two layers 10 m away from the sensor is already 35 cm. A visualization of this augmentation technique can be found in figure 4-9c.

Additionally to moving points individually, the whole object pointcloud can be shifted in distance. This technique is depicted in figure 4-9d. While rotating the pointcloud around its own axis does not change

The fifth augmentation technique that can be applied is the simluation of occlusions. As mentioned in section 2.1, an obstacle reflecting the sensor's laser pulse occludes the areas behind it. A traffic sign for example might occlude parts of a pedestrian, making a correct segmentation challenging. However it is important to especially detect VRU that are partly occluded, as they migh be standing behind a vehicle and are about to cross the road for example. Therefore, squares of random size and position created. All points within and behind this square are removed from the pointcloud. This approach takes inspriation from [SIN18]. As presented in section 2.10, random patches are placed onto images in order to hide random areas. Simulating occlusions is visualized in figure 4-9e.

Creating Semi-Artificial Samples

Before being able to insert objects into empty scenes, these objects have to be labeled and extracted from the recordings in a controlled environment. As hinted in section 4.3.1, labeling can be done by chosing a heigth treshold, labeling all points above. The points below the threshold are considered to be part of the ground. The threshold used has to be high enough to compensate irregularities of the ground. On the other hand it should be kept as low as possible to not cut off a large part of the objects of interest. In this thesis, it is chosen to be 10 cm.

In order to assign the labeled points to objects, a clustering algorithm can be used. In this work, mean-shift clustering is applied to all points belonging to the same class. As explained in section 2.11, mean-shift is able to cluster without providing the exact number of clusters to find. The pointcloud transformation presented in section 4.3.3 demands for a maximum distance sensed. If the objects recorded are occasionally leaving and entering the sensed area, the number of objects to find by clustering changes.

A structured database collecting all extracted pointclouds is created. An object from this database can be chosen and inserted into an empty scene. In this work, the object to insert is chosen by its distance. A random distance is sampled from a predefined distribution. The object in the database closest to this distance is selected.

Figure 4-11 depicts the extraction process. It can be seen that the points belonging to the two pedestrians (red) can automatically be labeled as they are the only points above ground-level. After clustering and extracting the objects, they can be inserted into the structured database.

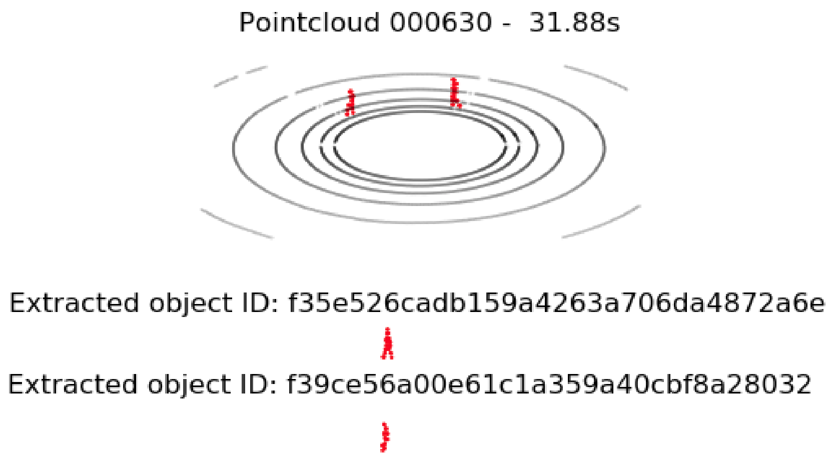

Pointcloud 000630 - 31.88s
Extracted object ID: f35e526cadb159a4263a706da4872a6e
Extracted object ID: f39ce56a00e61c1a359a40cbf8a28032

Fig. 4-11: Visualization of the object extraction. The two objects from the test track are labeled, clustered, extracted and the put into a structured database.

The selected object can be augmented using the above described techniques. Each technique receives a probability with which it is applied. Therefore, some objects might not get augmented at all while others are augmented with multiple techniques. In this work, augmentation techniques applied receive a probability of 50%.

After augmenting the selected object, it has to be checked if the insertion location is occupied. A simple 3D-bounding box is created around the object. If any point of the scene lies within this bounding box, the object is not inserted. If this is the case, the process of selection, augmentation and insertion check is repeated up to nine times. If the last insertion check fails, no object is inserted.

A picture of a semi-artificial pointcloud is depicted in figure 4-12. Additionally, a real pointcloud is shown. Both scenes seem very similar and presumably indistinguishable for the human eye underlying the motivation behind developing this technique. An image visualizing the bounding box can be found in figure 4-13.

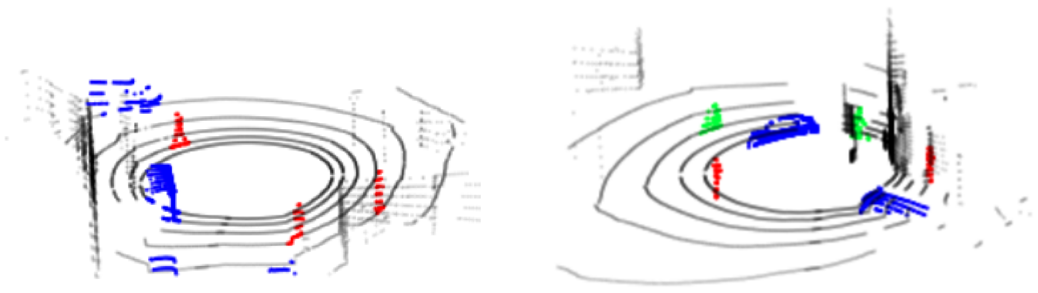


Fig. 4-12: Comparison of a real (left) and a semi-artificial scene (right).

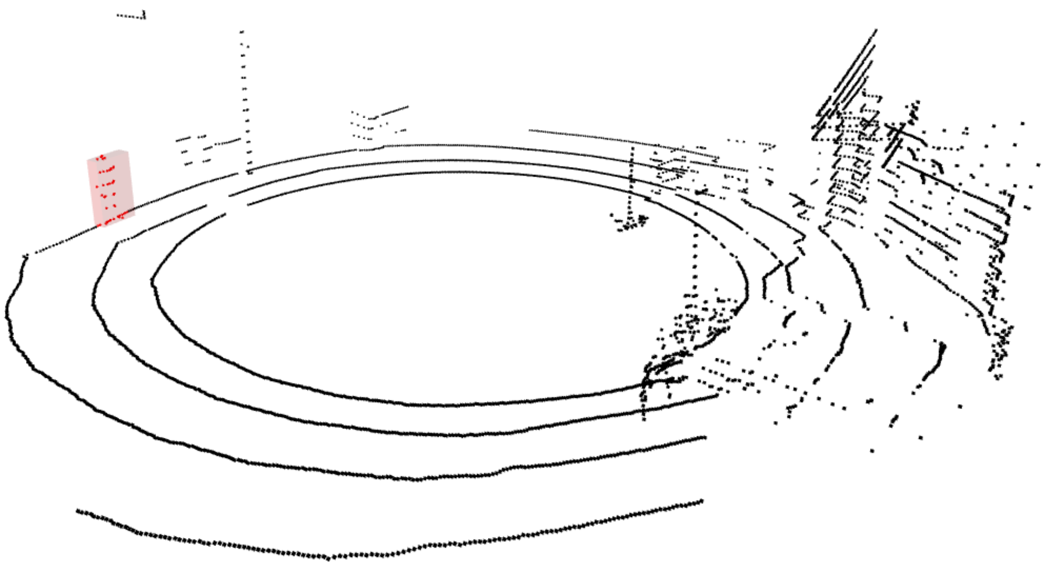


Fig. 4-13: Visualization of inserted objects with its bounding box. The bounding box represents the area occupied by an object.

Augmenting Semi-Artificial Samples

As mentioned before, the semi-artifical samples are not augmented in their pointcloud but their matrix representation. Two techniques are used in this work: Rotations and the addition of Gaussian noise.

Rotating the sample in its matrix representation can be realized by shifting its second dimension. As explained in section 4.3.3 the second dimension represents the azimuth angle. With respect to the azimuth resolution φ_{step}, shifting by n_{rot} indices along the second dimension corresponds to a pointcloud rotation of $n_{rot} \cdot \varphi_{step}$. The addition of Gaussian noise is straightforward, as it can simply be added to each value of the 3D-matrix representation.

It should be noted that the augmentation of empty scenes has been left out in this work. The reason behind this is that performing the two presented techniques on the semi-artificial sample should have the same effect as performing it onto both objects and empty scenes. Rotating the empty scene should have no effect before insertion if the objects are inserted at random angles. Therefore, it is sufficient to rotate the fused sample. The addition of Gaussian noise has the same effect no matter if applied to both object and empty scene or later of the semi-artificial sample, as the added noise is purely random. Difference would only arise if the noise distributions used would differ between objects and empty scenes. An additional difference in this thesis arises due to the application of the two techniques on the matrix representation. For example, the sample can only be rotated in steps of the discretization size of the azimuth dimension. However, the motivation behind augmenting the matrix has been explained in the beginning of the section.

Tuning the Augmentation Techniques

The techniques developed are parametrizable. An overview of all parameters is given in table 4-4. The augmentation techniques are mainly controlled by providing probability distributions. These distribution can directly be specified by well known distribution function. Additionally, a lower and upper boundary can be specified. Furthermore, it is also possible to use histograms in order to provide highly customized distributions used as basis to create random numbers. This enables the modification of the described techniques to a very high degree.

In addition to the probability functions, a few scalar values can be used to modify the creation of semi-artificial samples. The first one is a threshold for the obstacle detection. As described in section 4.3.2, before inserting an object, it has to be made sure that it is no overlap with an existing object in the scene. The threshold allows to accept a certain number of points within the object's bounding box before denoting an insertion position as occupied. An additional threshold used is the minimum number of points per object. This can be used to prevent the insertion of objects consisting of too few points, for example due to a faulty clustering in the object extraction process or to only a few points reflected by an object far away from the sensor. Furthermore, it is possible to specify the distribution of classes of objects selected for insertion.

Before inserting an object, the number of objects available for insertion gets limited to a certain class, sampled from the class probabilities specified.

technique	parameter	unit	num. values	prob. dist.
rotation	rotation angle	[°]	one	X
remove points	percentage of points to be removed	[-]	one	X
move single points	amount a point is shifted in azimuth and distance dircetion	[m]	two per point	X
shift object	distance-shift of the object	[m]	one	X
occlusions	size of the patch occluding the object	[m]	one	X
create semi-artificial samples	distance for object selection	[m]	one	X
	threshold for obstacle detection	[-]	one	
	minimum number of points per object	[-]	one	
	number of objects to insert	[-]	one	X
	distribution of classes of selected objects	[-]	one per class	
rotation	-	[-]	-	
gaussian noise	mean μ and standard deviation σ of the Gaussian noise	[-]	one per voxel	X

Tab. 4-4: Overview of the parameters defining the augmentation techniques. The last row indicates if the values are specified directly or via a probability distribution.

4.3.3 Pointcloud Transformation

A new matrix-representation of pointclouds has been developed for this work. Each LiDAR-pointcloud is transformed into a 3-dimensional matrix. The first two dimensions represent discretized distance and azimuth of the environment for each of the sensor's laser beams individually. The resulting 2D-grid maps are then concatenated in a third dimension. For the 16-layer sensor used in this thesis, this would lead to a third dimension of length 16 - if all beams are taken into account. As the LiDAR sensor's lasers have different elevation angles, this third dimension implicitly contains height information.

For the matrices' values, two different characteristics are examined: The number of points belonging to each voxel or the normalized intensity of these points. Using the intensity has the advantage of providing more information. However, using only the number of points would enable a wider use of this pointcloud representation. For example, a LiDAR sensor's intensities are very difficult to simulate. In a scenario where someone would like to merge simulation and real-world data, using the number of points is advantageous.

The matrix dimensions' size depends on the discretization of the pointcloud. A maximum sensable distance and a spatial resolution has to be chosen, leading to a first dimension of size:

$$D_1 = \frac{r_{max}}{r_{step}} \qquad \text{Eq. 4-6}$$

For the second dimension, only the angular resolution is needed to compute the dimension's size. The size of the second dimension therefore is:

$$D_2 = \frac{360^\circ}{\varphi_{step}} \qquad \text{Eq. 4-7}$$

In scenarios where no 360° perception is demanded, the second dimension can be modified accordingly.

For each LiDAR point with coordinates P (see equation 4-5) within the range r_{max}, the corresponding voxel characterized by d_1, d_2 and d_3 can be computed as follows:

$$d_1 = \left\lfloor \frac{r}{r_{step}} \right\rfloor, \qquad d_2 = \left\lfloor \frac{\varphi}{\varphi_{step}} \right\rfloor, \qquad d_3 = n_{laser} \qquad \text{Eq. 4-8}$$

The transformation performed on each point is computationally cheap as φ, r and n_{laser} are directly provided by the LiDAR sensor.

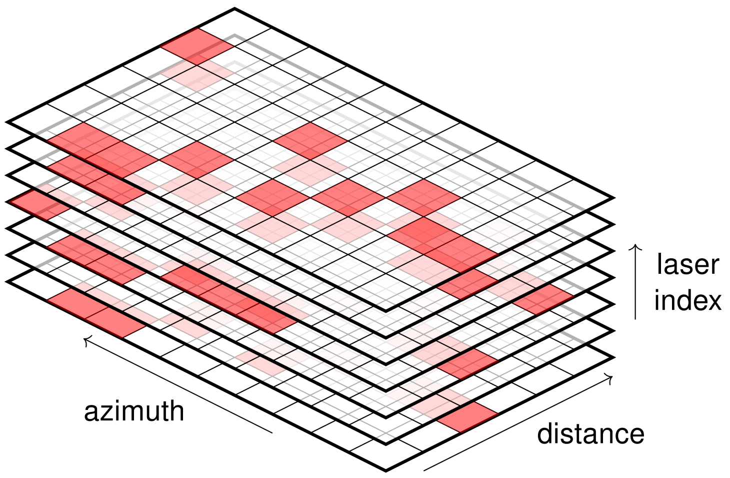


Fig. 4-14: Visualization of the matrix that serves as input for the convolutional neural network. Each point of the pointcloud gets assigned to its voxel in the matrix by computing discretized azimuth and distance as well as laser index. Cells marked red are non-zero, indicating the sparsity of the matrix. The matrix values can either be the number of points assigned to that voxel or the normalized intensity sum.

4.3.4 Neural Network for Semantic Segmentation

Multiple CNN architectures have been examined with respect to the requirements raised in section 4.2. Different variations of PointNet, ResNet and SqueezeSeg as well as a simplistic architecture only consisting of a few convolutional layers have been tested.

As mentioned in section 4.2.3, the search for a suitable network architecture was focused on convolutional neural network. SqueezeSeg presented in section 2.7 was chosen as basis as it is reaching state of the art performance for semantic segmentation of pointclouds. However, SqueezeSeg overfits highly on the data used in this thesis. A possible reason for that could be that SqueezeSeg is designed for a sensor with double the number of layers. As pointed out in section 2.6, the capacity could be to high for operating on pointclouds containing with only 16 laser diodes.

In order to tackle the overfitting problem, the model's capacity was reduced by decreasing the number of layers and modifiying kernel sizes the network contains. The topology developed in this work is visualized in figure A-9. When comparing the neural network used in this thesis to SqueezeSeg, the main difference is a reduction of layers. Instead of the 15 layers in SqueezeSeg, only seven are used. For example, the third downsampling step in SqueezeSeg is completely skipped. The network used in this work only gets downsampled twice: Once by the first convolutional layer with a stride of two and then once again by a maxpooling layer. Additionally, a single convolutional layer in parallel to all other layers has lead to an additional increase in performance.

A tabular overview about the layers used is found in table 4-5. It can be seen that in each convolutional layer, a high number of filters has been used. Though, the kernel dimension are kept low with sizes of 3x3 and 1x1, a central property of SqueezeSeg, as explained in section 2.7. ReLU is used as neuronal activation function.

In order to reduce the risk of overfitting, a dropout layer is used directly before performing the softmax operation. 50% of all neurons are dropped during the training period.

layer name	input	type	functionality
conv1	input	convolution	64 filters, 3x3 kernel, stride=2
conv1skip	input	convolution	64 filters, 3x3 kernel, stride=1
pool1	conv1	pooling	maxpool, size=3, stride=2
fire1	pool1	fire	16 1x1 squeeze filters, 64 1x1 + 64 3x3 expansion filters
fire2	fire1	fire	16 1x1 squeeze filters, 64 1x1 + 64 3x3 expansion filters
fire3	fire2	deconv	16 1x1 squeeze filters, 32 1x1 + 32 3x3 expansion filters
fire3fuse	sum	fire2	fire3 + conv1
fire4	fire3fuse	deconv	16 1x1 squeeze filters, 32 1x1 + 32 3x3 expansion filters
fire4fuse	fire4	sum	fire4 + conv1skip
dropout	fire4fuse	dropout	50% keep probabilty, only active during training
softmax	dropout	softmax	softmax function for classification

Tab. 4-5: Description of the layers used to build the neural network developed in this work. A visualization is found in figure A-9

4.3.5 Evaluation

Annotation of the Evaluation Data

As described in the previous section, drone videos were taken during the recording of the evaluation data set in order to assist the labeling process. As mentioned in section 2.9, annotating pointclouds can be more difficult than annotating images for a human. Looking at pointclouds, it is often not easy to differentiate between a pedestrian or pole for example. In contrast to that, it is much easier to differentiate between them by looking at an image. Therefore, this work proposes to use drone images assisting the pointcloud labeling process.

The key idea in drone assisted labeling of pointclouds is to label the drone images and transfer the label to the pointclouds. This can be done by fusing the pointcloud with the drone image. To do so, the pointclouds' coordinates in meters have to be transformed into their pixel representation and the origin of the pointcloud has to be detected within the image.

In order to label the drone images, a labeling tool has been developed within this thesis. The tool is based on the City Scapes labeling tool mentioned in section 2.9. This tool provides a basic interface for image annotation. In addition to modifications necessary fit the tool to enable easy annotation of the drone images used in this work, a neural network suggesting annotations has been developed and included into the tool. After labeling a few drone images by hand, a neural network using the resnet-like architecture [POH17] presented in section 2.7 has been trained. As only a few labeld images are used for training, the images are augmented by rotations and flipping, as suggested in 2.10. A visualization of the labeling process can be found in figure 4-16.

The trained resnet-like network performs predictions on all images the human annotator labels. Due to the small training data set, the predictions might be inaccurate. The human annotator has to correct the labels in order to ensure accurate labeling. If correcting the network's suggestions is faster than labeling the image as a whole, the labeling cost can be reduced. Additionally, the model can be retrained after labeling additional samples, presumably leading to better predictions. This way, the labeling time might be reduced further successively.

Evaluation Metrics

As suggested in section 4.2.5, multiple evaluation metrics are used. This work evaluates the concept for pointcloud augmentation using three of the most common metrics used for semantic segmentation, namely IoU, precision and recall. These metrics are already described in section 2.12. Intersection over Union (IoU) is used as the main measure of performance. Additionally, precision and recall are examined to get a better understanding of the networks performance.

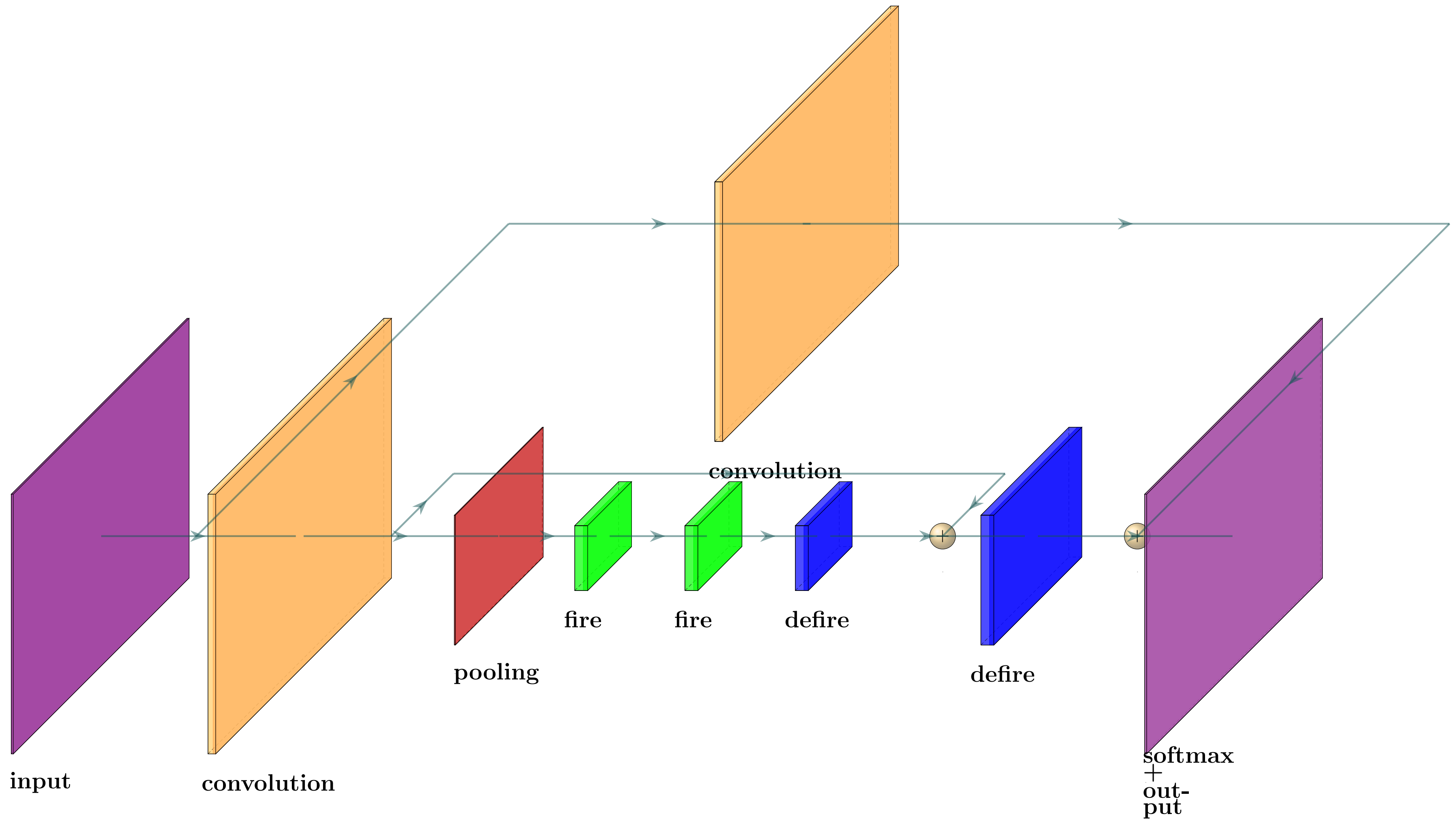
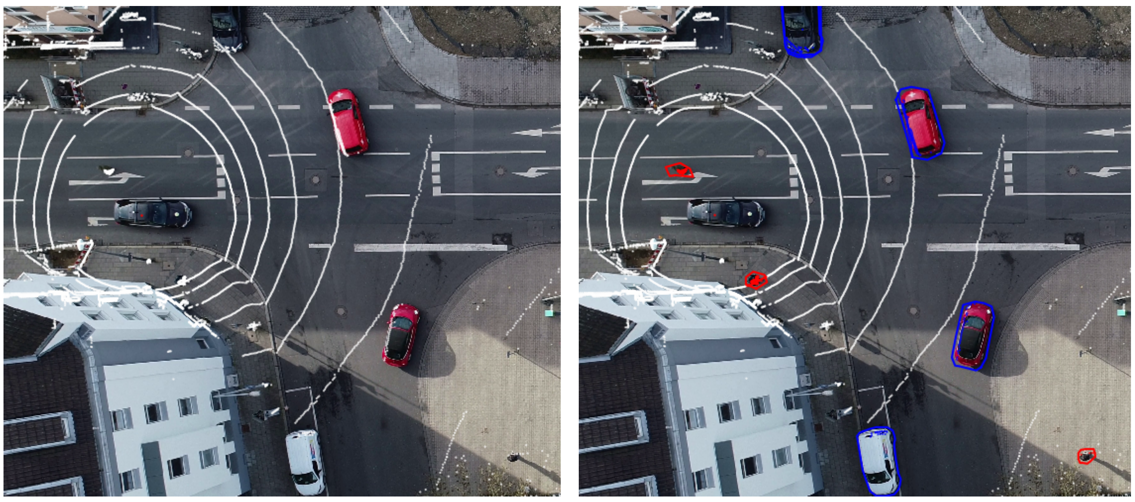


Fig. 4-15: Architecture of the convolutional neural network used in this thesis. The firemodules used are taken from the SqueezeSeg architecture. Two skip connections are used.


a) Fusing drone images with pointclouds.

b) Labeling the fused pointclouds using the labeld drone image.

Fig. 4-16: Visualization of the drone assisted labeling of pointclouds. a) shows the fusion of drone image and pointclouds by transforming the point coordinates in meters to the pixel domain. b) depicts the labeling process. The polygons indicate labeled objects, the color indicates their class (red: pedestrian, blue: vehicle). All points within a polygon are labeled. The color of a point indicates its assigned class.

5 Experimental Design

A concept for data augmentation has been presented in the previous chapter. Aiming at answering the research question raised in 3, this concept has to be evaluated. In this chapter, four experiments are presented. The first three experiments try to answer the three secondary issues proposed in chapter 3. For these experiments, only one class is taken into account. The class *pedestrian* was chosen due to the high availability of both extracted pedestrians as well as scenes without them. The fourth experiment is performed to study how the results obtained considering one class can be compared to data sets containing multiple classes. The additional class *vehicle* is introduced.

The subsequent experiment focuses on the influence of traditional augmentation techniques. These techniques are performed for the data sets of the first experiment, so that data sets without the application of traditional augmentation techniques can be compared to those where they are applied. The third experiment is performed to compare manually annotated pointclouds to recordings of real road traffic. A manually labeled data set is augmented by both traditional techniques and the creation of semi-artificial samples.

As explained in section 2.6, evaluation metrics on the training data do not offer information about the networks generalization. Therefore, all metrics examined in chapters 5 and 6 are the validation metrics. Tu further improve the validity of the results, all tests presented within the experiments are performed three times and the resulting metrics are averaged. The results obtained by each measurement can be found in the appendix.

In all four experiments, the number of points per voxel has been chosen as characteristic over the intensity. As mentioned in section 4.3.3, a possible alternative is the average, maximum or accumulated intensity. However, using the intensity did not lead to a meaningful performance increase during the development phase. Due to the advantages presented in section 4.3.3, the number of points has been chosen. A maximum distance of 10 m is considered. The distance resolution is 10 cm. The angular coordinate has been divided into 176 bins, leading to a resolution of 2.05°. The full number of LiDAR layers was used. This leads to an input matrix of size $100 \times 176 \times 16$. The parameters defining the pointcloud representation are chosen based on qualitative results, for example by looking at the visualization of the semantic segmentation. An image showing a visualized pointcloud used to for qualitative evaluation can be found in the appendix in figures A-2 and A-3. As the focus of this work lies on augmentation, quantitative comparisons are not depicted.

5.1 Experiment 1: Creation of Semi-Artificial Samples

The first experiment aims at studying the creation process of semi-artificial samples. It examines following question raised in chapter 3:

How does the performance of a CNN trained on semi-artifical samples compare to state-of-the-art performances of CNNs trained on manually labeled pointclouds from recorded traffic scences?

In order to focus on the creation of semi-artificial samples, traditional augmentation techniques are not considered in this experiment. Following hypothesis is proposed: The number of objects and the number of empty scenes both have an impact on the quality of the created data set. It is assumed that a higher availability of at least either one of them leads to a higher performance of a neural network trained on the data.

To test this hypothesis, multiple data sets are created, each one with a different number of empty scenes or objects available. First, only the number of objects available is varied while the number of empty scenes is kept constant. Following, the number of empty scenes is varied while the number of available objects stays fixed. An overview of the variations tested is depicted in figure 5-1.

Test Number	number of empty scenes used	number of objects available for insertion
1	10000	100
2	10000	500
3	10000	3000
4	10000	30000
5	100	30000
6	1000	30000
7	5000	30000
8	10000	30000

Tab. 5-1: Overview of the parameters varied in experiment 1.

10000 semi-artificial samples are created for each data set. During development, this data set size has proven to be suitable for training. A comparison of performances for different data set sizes can be found in the appendix. The parameters influencing the creation of semi-artificial samples are not varied in this experiment. The values used for the parameters presented in section 4.3.2 can be found in table 5-2.

A neural network is trained for each data set presented in table 5-1. The model's architecture is found in section 4.3.4. Each training is performed for 100 epochs. This value was chosen as the performance typically converged before reaching this number of epochs. After training,

parameter	Value	lower bound	upper bound
distance for object selection	$\sim U(1,10)$	-	-
threshold for obstacle detection	60	-	-
minimum number of points per object	5	-	-
number of objects to insert	$\sim N(\mu=1,\sigma=0.5)$	0	10
distribution of classes of selected objects	$p_{pedestrian}=1$	-	-

Tab. 5-2: Overview of the parameters to tune the augmentation in experiment 1.

the performance of the epoch showing the highest IoU is selected and IoU, precision and recall compared for this epoch. It should be noted that the epoch showing maximum IoU is not always the epoch with maximum precision or recall. But as the metrics are all based on the same confusion matrix (see section 2.12), choosing different epochs would diminish comparability.

The data set consisting of manually annotated pointclouds presented in section 4.3.1 is used as validation data set.

The results obtained in this experiment can be compared among themselves and to the performance achieved by other state-of-the-art networks. However, the comparison to other networks has to be performed with care. Different data sets and different pointcloud representations as network input might impair the comparability. Nevertheless, the comparison to other networks can be used for an approximate assessment.

5.2 Experiment 2: Traditional Augmentation Techniques

The second experiment focuses on the traditional augmentation techniques developed in this work. The motivation is to study the following question, which was raised in chapter 3:

In which way is the performance of a CNN influenced by traditional augmentation techniques applied to its training pointclouds?

In order to find an answer to this question, the augmentation of objects available for insertion and the augmentation of semi-artificial samples are examined. The creation process of semi-artificial samples is based on the same parameters used in the previous experiment. By doing so, all tests performed can be evaluated to the corresponding non-augmented data set created for experiment 1. Following hypothesis is assumed: Data sets consisting of 10000 samples benefit from traditional augmentation techniques, regardless of the number of empty scenes or objects available. Furthermore, it estimated that this benefit decreases with a higher number of empty scenes and objects available, because the baseline performance of the non-augmented data rises.

A list of all tests regarding the augmentation of inserted objects is presented in figure 5-4. Point removal, spatial noise and distance shift are each evaluated once individually and once combined with rotating the object. Regarding augmentation by occlusion, two parameter choices are evaluated: One with a small and another with a large patch size. They will be referred to as small and large occlusion respectively. Both are evaluated individually as well as combined with rotation. Two more techniques studied are the application of rotations alone and the combination of all presented augmentation techniques for objects. All techniques are performed for the same variations of the number of available objects used for experiment 1, i.e. 100, 500, 3000 and 30000 objects.

To augment an object by rotating it, a random angle has to be sampled. For this experiment, a uniform distribution between 0° and 360° is used to sample the angle. For the other techniques, namely point removal, spatial noise, distance shift and occlusion, Gaussian distributions with upper and lower bounds are chosen to sample the relevant parameters. The exact distributions are found in table 5-3.

The two augmentation techniques performed on the semi-artificial samples are evaluated individually as well as combined. The evaluation is done for the same variations of the number of empty scenes used for experiment 1, i.e. 100, 1000, 5000 and 10000. The number of available objects is kept at 30000 to not introduce too many possible variations. The Gaussian distribution used to add noise is depicted with all other augmentation parameters in table 5-3.

technique	**parameter**	**Value**	**lower bound**	**upper bound**
rotation	rotation angle	U(0,2π)	-	-
remove points	percentage of points to be removed	$\sim N(\mu = 0.1, \sigma = 0.20)$	0	0.6
move single points	amount a point is shifted in azimuth and distance direction	$\sim N(\mu = 0.0, \sigma = 0.07)$	-0.1	0.1
shift object	distance-shift of the object	$\sim N(\mu = 0.0, \sigma = 0.40)$	-2	2
small occlusions	size of the patch occluding the object	$\sim N(\mu = 0.2, \sigma = 0.30)$	0.1	0.5
large occlusions	size of the patch occluding the object	$\sim N(\mu = 0.5, \sigma = 0.30)$	0.1	0.9
create semi-artificial samples	distance for object selection	U(1,10)	-	-
	threshold for obstacle detection	60	-	-
	minimum number of points per object	5	-	-
	number of objects to insert	$\sim N(\mu = 1, \sigma = 0.5)$	0	10
	distribution of classes of selected objects	$p_{pedestrian} = 1$	-	-
rotation	-	-	-	-
Gaussian noise	mean and standard deviation	$\sigma = 0,\ \mu = 5$	-	-

Tab. 5-3: Overview of the parameters defining the augmentation techniques. The last row indicates if the values are specified directly or via a probability distribution.

The same method to receive evaluation metrics as in experiment 1 is used. For each data set created, a model is trained for 100 epochs. IoU, precision and recall are noted for epoch showing highest IoU. The results are compared to the non-augmented data sets created in experiment 1. The data set consisting of manually annotated pointclouds presented in section 4.3.1 is used as validation data set. This means, the augmentation techniques applied using a certain number of available objects are compared to the data set using the same number of objects. Comparing the techniques with respect to a different number of empty scenes used happens analogous to that. The same data set as in experiment 1 is used as validation set.

	number of empty scenes used	number of objects available for insertion	augmentation techniques
1	10000	100	rotation
2	10000	100	remove points
3	10000	100	remove points, rotation
4	10000	100	spatial noise
5	10000	100	spatial noise, rotation
6	10000	100	shift object
7	10000	100	shift object, rotation
8	10000	100	small occlusion
9	10000	100	small occlusion, rotation
10	10000	100	large occlusion
11	10000	100	large occlusion, rotation
12	10000	100	all techniques
13	10000	500	rotation
14	10000	500	remove points
15	10000	500	remove points, rotation
16	10000	500	spatial noise
17	10000	500	spatial noise, rotation
18	10000	500	shift object
19	10000	500	shift object, rotation
20	10000	500	small occlusion
21	10000	500	small occlusion, rotation
22	10000	500	large occlusion
23	10000	500	large occlusion, rotation
24	10000	500	all techniques
25	10000	3000	rotation
26	10000	3000	remove points
27	10000	3000	remove points, rotation
28	10000	3000	spatial noise
29	10000	3000	spatial noise, rotation
30	10000	3000	shift object
31	10000	3000	shift object, rotation
32	10000	3000	small occlusion
33	10000	3000	small occlusion, rotation
34	10000	3000	large occlusion
35	10000	3000	large occlusion, rotation
36	10000	3000	all techniques
37	10000	30000	rotation
38	10000	30000	remove points
39	10000	30000	remove points, rotation
40	10000	30000	spatial noise
41	10000	30000	spatial noise, rotation
42	10000	30000	shift object
43	10000	30000	shift object, rotation
44	10000	30000	small occlusion
45	10000	30000	small occlusion, rotation
46	10000	30000	large occlusion
47	10000	30000	large occlusion, rotation
48	10000	30000	all techniques

Tab. 5-4: Overview of experiment 2: Variation of the number of objects available

	number of empty scenes used	number of objects samples created	augmentation techniques
49	100	30000	rotation
50	100	30000	noise
51	100	30000	rotation, noise
52	1000	30000	rotation
53	1000	30000	noise
54	1000	30000	rotation, noise
55	5000	30000	rotation
56	5000	30000	noise
57	5000	30000	rotation, noise
58	10000	30000	rotation
59	10000	30000	noise
60	10000	30000	rotation, noise

Tab. 5-5: Overview of experiment 2: Variation of the number of empty scenes used

5.3 Experiment 3: Comparing Semi-Artificial and Real Data Sets

After examining the developed augmentation techniques in experiments 1 and 2, experiment 3 aims to find out how distinguishable semi-artificial samples are from real ones. This is done aiming to answer following question presented in chapter 3:

To which extend do the developed augmentation techniques lead to a supplement or even substitution of manually annotated pointclouds from recorded traffic scenes?

A first tendency about the answer of this question can be observed in the results of experiment 1. If the experiment shows that models trained on the created data sets can reach reasonable performance compared to the state of the art, the hypothesis assumed is that semi-artificial samples can be used both as a supplement or substitution for real samples. For further validation, a direct comparison between real and semi-artificial samples is done in this experiment.

Manually annotated pointclouds of road traffic are needed for the comparison. However, the only manually labeled data set is the one used for validation in the previous experiment. Therefore, this data set has to be split into a training and a validation set. This means that training and validation samples are possibly highly similar. It has to be noted that this approach diminishes the reliability of the obtained results. There is a possibility that good evaluation metrics are caused by an overfitting model. If training and validation data are very similar, overfitting might not always be detected. Due to the high cost of obtaining labeled pointclouds, which has been highlighted throughout this thesis, obtaining more labeled pointclouds of different locations was not feasible in this work.

The results obtained in this experiment should therefore be handled with care. However, they are suitable to identify tendencies and potentials.

First, the potential of semi-artificial samples as supplement for real data is studies. To obtain a baseline, a network is trained on the manually labeled data without augmentation. Then, labeled objects from the labeled scenes are extracted. This leads to empty scenes and variety objects that can be reinserted. Two data sets are created using the split data: The objects originally positioned in the scenes are augmented and reinserted. Due to the augmentation, the objects can be inserted into different scenes and at different locations than their original state. The second data set is created by inserting the objects extracted from the controlled environment. Similar to the previous experiments, the size of the two data sets is set to 10000 samples. The data creation parameters a equivalent to experiment 1, i.e. depicted in table 5-2.

The models trained on these data sets are trained for 100 epochs. However, the non augmented data set is trained for 500 epochs. As it consist of much less samples, fewer weight modification steps can be done. The model converges more slowly and therefore needs to be trained for a longer time. Obtaining the corresponding metrics is done analogously to experiment 1 and 2. Obtaining the metrics is done as in the previous experiments.

After examining the semi-artificial samples as supplement, their potential to fully substitute manually annotated data is evaluated. In order to do so, the empty scenes obtained from splitting the annotated pointclouds can be reused. Objects extracted from the controlled environment can be inserted at the exact same positions as the original objects. This way, semi-artificial scenes are created which have an almost identical sample as within the real data. The only difference is that the original objects are swapped by objects recorded in a controlled environment. For evaluation, the IoU of both models is compared for the whole training period, i.e. for every epoch.

5.4 Experiment 4: Data Sets with Multiple Samples

The fourth experiment aims at transferring the previous results to the multi-class domain. While the previous experiments consider pedestrians as the only class, data sets containing pedestrians and vehicles are created. It is assumed that the results obtained in the previous experiments can be transferred to other classes as well. Though an-in depth evaluation is outside of the scope of this work, an outlook can be given. Accordingly, the results should only be used to describe tendencies and potentials.

A semi-artificial data set containing all three classes is created. The data set size is expanded to 30000 samples through augmentation. The model is trained on this data set for 500 epochs. These parameters are enlarged as it is assumed that the segmentation task is more challenging if more classes have to be predicted. Again, IoU, precision and recall are noted for each trained epoch. 30000 empty scenes and all objects extracted from the controlled environments are used. An overview about the available of objects per class is found in the concept chapter in section 4.3.1. The probability of inserting a pedestrian or a vehicle is set to equally 50%.

As bicyclist have been recorded on the test track, it would be possible to include them in this experiment. However, due to the small amount of bicyclists in the evaluation data set, bicyclist have been left out.

6 Results

This chapter evaluates the results of the experiments presented in chapter 5. First, the results of each experiment are examined individually. Then, section 6.5 summarizes the results and tries to answer the questions raised chapter 3.

6.1 Experiment 1: Creation of Semi-Artificial Samples

This section aims at testing the hypothesis that the performance of a neural network trained on the semi-artificial data set increases if the number of empty scenes or objects available is increased. Figure 6-1 shows the results for the tests varying the number of objects available. Figure 6-2 shows the results for the tests varying the number of empty scenes available. The results are visualized using bar charts. An individual chart is created for IoU, precision and recall respectively. Each test listed in table 5-1 is represented by its own bar.

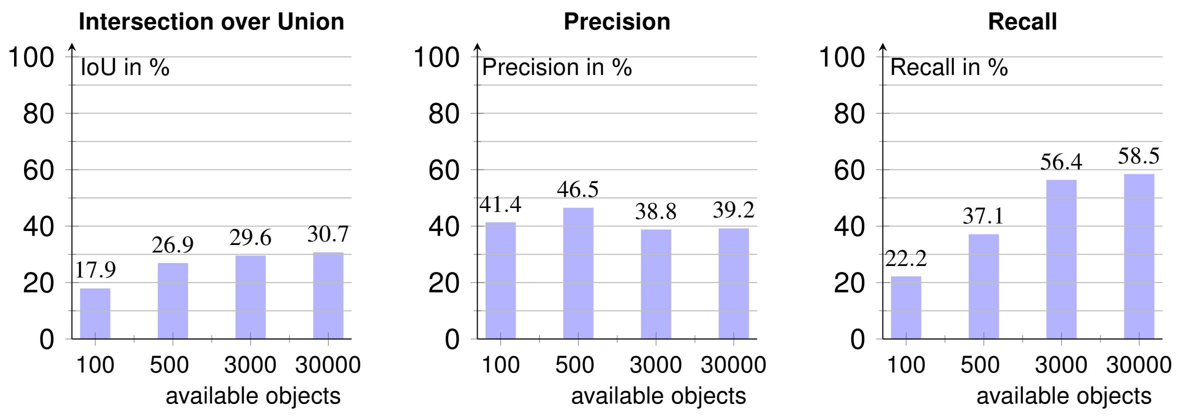


Fig. 6-1: Results of Experiment 1: Variation of the number of objects available. Three diagrams are depicted that show the evaluation of IoU (left), precision (center) and recall (right).

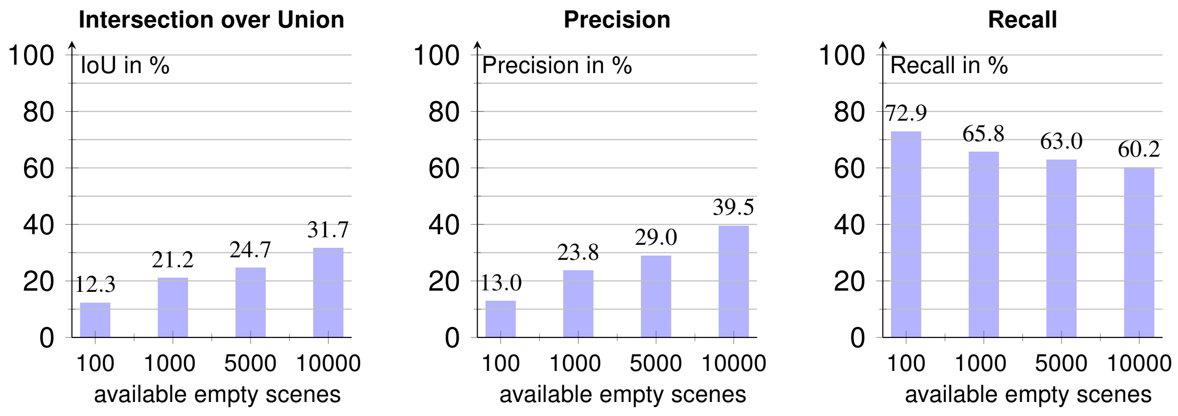


Fig. 6-2: Results of experiment 1: Variation of the number of empty scenes available. Three diagrams are depicted that show the evaluation of IoU (left), precision (center) and recall (right).

As described in section 5.1, the results obtained are compared to performances of a state-of-the-art network. As presented in section 4.3.4, the network developed in this thesis is based on the SqueezeSeg architecture. Therefore, the results are compared to the performances of SqueezeSeg. The network's metrics are measured on the KITTI data set. For pedestrians, the original SqueezeSeg achieves an IoU of 22.8%, a precision of 52.9% and a recall of 28.6% [WU17]. As a different data set and a different pointcloud representation are used, these values are only used to evaluate if the performances measured in this work are within the scope of state of the art performances. A quantitative comparison is not suitable. Especially the distance considered, i.e. 10 m in this work, presumably has a high influence on the performance.

Regarding figure 6-1, it can be seen that IoU and recall both increase when increasing the number of available objects. Using 30000 objects, an IoU of 30.7 % is measured. The recall is 58.5 %. As both values lie above the ones of SqueezeSeg, we assume that it is possible to achieve an IoU and a recall comparable to the state of the art. The maximum precision, which is achieved using 500 available objects, is 46.5 % and therefore 6.4 percentage points (pp) below SqueezeSeg's precision on KITTI.

The benefit of using 30000 objects for insertion seems small compared to using only 3000 objects. Using ten times the number of available objects only leads to an increase of 1.1pp regarding IoU and 2.1pp regarding recall. Precision increase by 0.4pp, staying under the precision achieved using 500 available objects.

Figure 6-2 shows that using more empty scenes for the data set creation lead to an increasing IoU. However, in contrast to varying the number of objects, the recall decreases while the precision increases. Another difference is the fact that using more empty scenes always leads to a noticeable increase in IoU for the examined tests.

It should be noted that the bars belonging to 30000 available objects in figure 6-1 have the same number of both available objects as well as empty scenes as the bars corresponding to 10000 available scenes in figure 6-2. Therefore, the performances should be almost equal. Small changes can arise because the data sets are created under the same premises but are not identical.

It seems reasonable that using a small number of available objects with a high number of empty scenes leads to a low recall. If the training data has many differing examples of empty scenes and only a few examples of objects, it might learn the exact shapes of the objects available, considering all other shapes as background. This can lead to a high number of false negatives, causing a low recall. The same hypothesis can be applied to to a high number of available objects with a low number of empty scenes used. It is possible that the network learns the exact shapes of the background and therefore tending to classify an unknown shape as object. This leads to a high number of false positive and thus to a lower precision.

The hypothesis proposed in section 5.1 therefore has to be answered with respect to the metrics studied. The data suggests that an increase in empty scenes seems to lead to an increasing

IoU and precision. According to the results, an increase in available objects seems to lead to an increasing IoU and recall.

6.2 Experiment 2: Traditional Augmentation Techniques

As described in section 5.2, the CNN is trained on each data set listed in tables 5-4 and 5-5. IoU, precision and recall are evaluated, measuring the change with respect to the non augmented data set in percentage points.

Augmentation of Inserted Objects

Again, bar charts are chosen to visualize the results. Two bars for each augmentation technique are plotted on the x-axis: The first one representing the augmentation technique alone and the latter one representing an additional rotation, as explained in section 5.2. The metrics of all data sets containing the same number of objects available for insertion are visualized in one diagram. The results regarding IoU are shown in figure 6-3. Corresponding results regarding precision and recall can be found in figure 6-4 and 6-5 respectively. Due to the high number of tests performed in this experiment, the results are structured with respect to the evaluation metrics.

Intersection over Union Regarding the change in IoU in figure 6-3, it can be seen that data sets created with only 100 objects available for insertion benefit highly from augmentation techniques. Every augmentation technique causes an increase in IoU. However, removing random points does not show to be very beneficial. An additional benefit is achieved when combining the techniques with an augmentation by rotation. However, the combined techniques should be compared to the IoU achieved by applying rotation as only augmentation technique instead of to the baseline.

The highest increase in IoU is seen when combining small occlusions with a rotation of the object: The increase of 18.61pp more than doubles the IoU of the non-augmented data set. The resulting IoU of 36.5% is even higher than the performance achieved using all objects.

Another notable observation is that applying an augmentation technique in combination with a rotation does not lead to a performance gain equivalent or close to the sum of both individually achieved gains. Additionally, it is worth to highlight that applying all augmentation techniques together does not lead to the highest performance gain.

Augmentation techniques applied to data sets with 500 available objects lead to an increase in IoU for every technique examined. Compared to the data sets with 100 available objects, this increase is however of smaller size. This fact seems intuitive, as the baseline IoU is higher. The highest increase is caused by applying small occlusion. In contrast to the data sets with 100 available objects, this increase is achieved without combining the occlusion with a rotation of the inserted object. The benefit of combining rotations with other techniques seems to vanish in general for the data sets with 500 objects available. Spatial noise is the only technique benefiting from the combination, but only by 2.6pp. A possible explanation is that 500 objects presumably cover more azimuth angles than 100 objects. This might lead to a lower benefit of rotating the objects. Again, it should be emphasized that combining all augmentation techniques does not seem to beneficial with respect to the performance change obtained by individual techniques.

Regarding the results on the data sets with 3000 available objects, the benefit of most augmentation techniques fades or is even reversed. Applying only rotations, removing points or simulating large occlusions leads to relatively small IoU gains. All other techniques show negligible or even negative IoU changes. With 3.8pp, applying the spatial noise technique leads to the highest decrease.

The application of the examined techniques to data sets created with 30000 available objects seems to impair the IoU of a network trained on them. Besides the technique of removing points, all techniques applied lead to a decrease of the IoU. Removing points does not seem to have any impact in these tests. Additionally, the data suggests that the combination of techniques with rotations impairs the IoU even further. With around 6pp for rotations combined with removing points, spatial noise or the distance shift, the size of the decrease is relatively large.

Precision When examining the IoU, the benefit of the augmentation techniques applied to objects decreases with increasing number of available objects, in the end even impairing the performance. The same trend can be observed when studying the precision in figure 6-4.

Regarding the data sets created using 100 available objects, most augmentation techniques show a high increase in precision. Though, the removal of points, applying small occlusions and applying all techniques seem to have no impact on the precision. The largest increase is seen when combining small occlusions with a rotation of the objects, analogous to the IoU.

While data sets based on 500 available objects still seem to benefit from augmentation of the inserted objects regarding IoU, most techniques do not lead to an increase in precision. Applying small occlusions or distance seem to be beneficial, but any other technique either shows very small increases or even decreases. The tendency, that the traditional augmentation techniques start to impair the performance can also be seen for the data sets created using 3000 and 30000 available objects respectively.

Recall The data suggests that the recall is the only metric for which the hypothesis formulated in 5.2 could hold true. Examining figure 6-5, it can be seen that all augmentation techniques correlate with an increasing recall. As assumed, the benefit decreases with an increasing number of available objects. The negative change in recall when applying the point removal is so small that it can be neglected. When comparing the plots for 3000 and 30000 available objects, it can not be seen whether the augmentation techniques generally perform better on one of those data sets.

In contrast to IoU and recall, applying all augmentation techniques together leads to the highest increase in recall for each data set. Additionally, for 30000 available objects, the introduction of small occlusions leads to the same increase in recall.

After performing the test examining the augmentation of inserted objects, the hypothesis postulated in section 5.2 is supported by the data with respect to the recall. Data sets with augmented objects show a higher recall than non augmented data sets. The benefit decreases with increasing number of available objects. Regarding IoU and precision, the hypothesis only holds true for data sets with a very small number of available object. The assumption that data sets with a higher number available objects benefit as well is rejected.

Augmentation of Semi-Artificial Samples

Analogous to the previous visualizations, bar charts are used as well to evaluate the augmentation of semi-artificial samples. The results are shown in figure 6-6. Three diagrams are plotted, depicting the results for the change in IoU, precision and recall respectively. A bar for each augmentation technique is created, i.e. rotation, adding noise and combining both techniques. Groups with respect to the number of available empty scenes are created.

The results of the augmentation techniques applied to semi-artificial samples do no allow many conclusions. The addition of Gaussian noise leads to changes that seem arbitrary regarding the metrics and number of empty scenes available. For example, no explanation has been found for why the data set created with 1000 available scenes hardly benefits from rotation, shows a decrease in in all metrics for the application of noise, but exhibits an increase in IoU and precision when both techniques are combined.

It is noticeable that all three examined techniques cause their highest increase in IoU and precision for data sets created with 5000 available objects.

In summary, regarding the augmentation of semi-artificial samples, the hypothesis postulated in section 5.2 can neither be verified nor rejected.

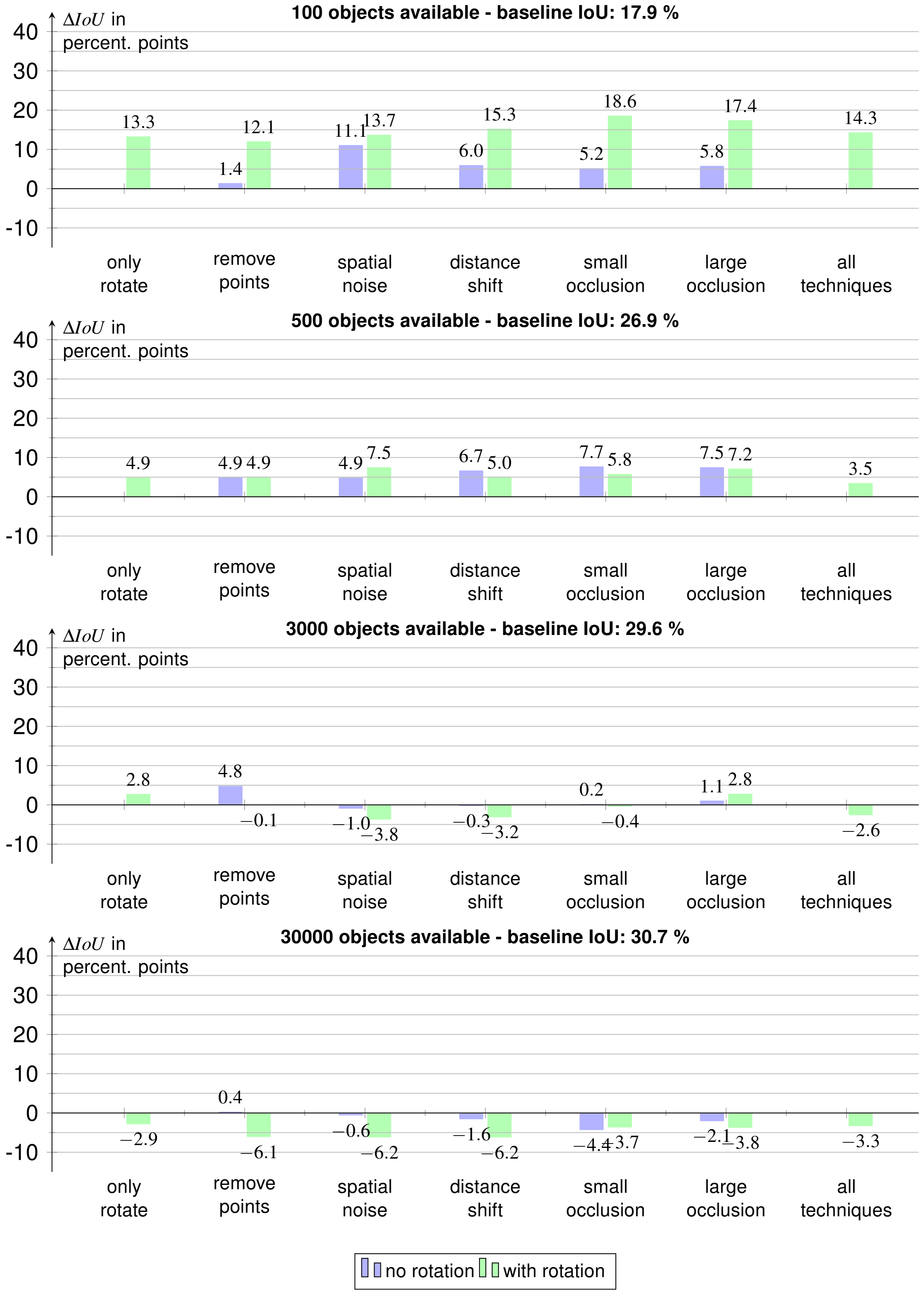


Fig. 6-3: Results of Experiment 2regarding IoU for varying the number of objects available for insertion.

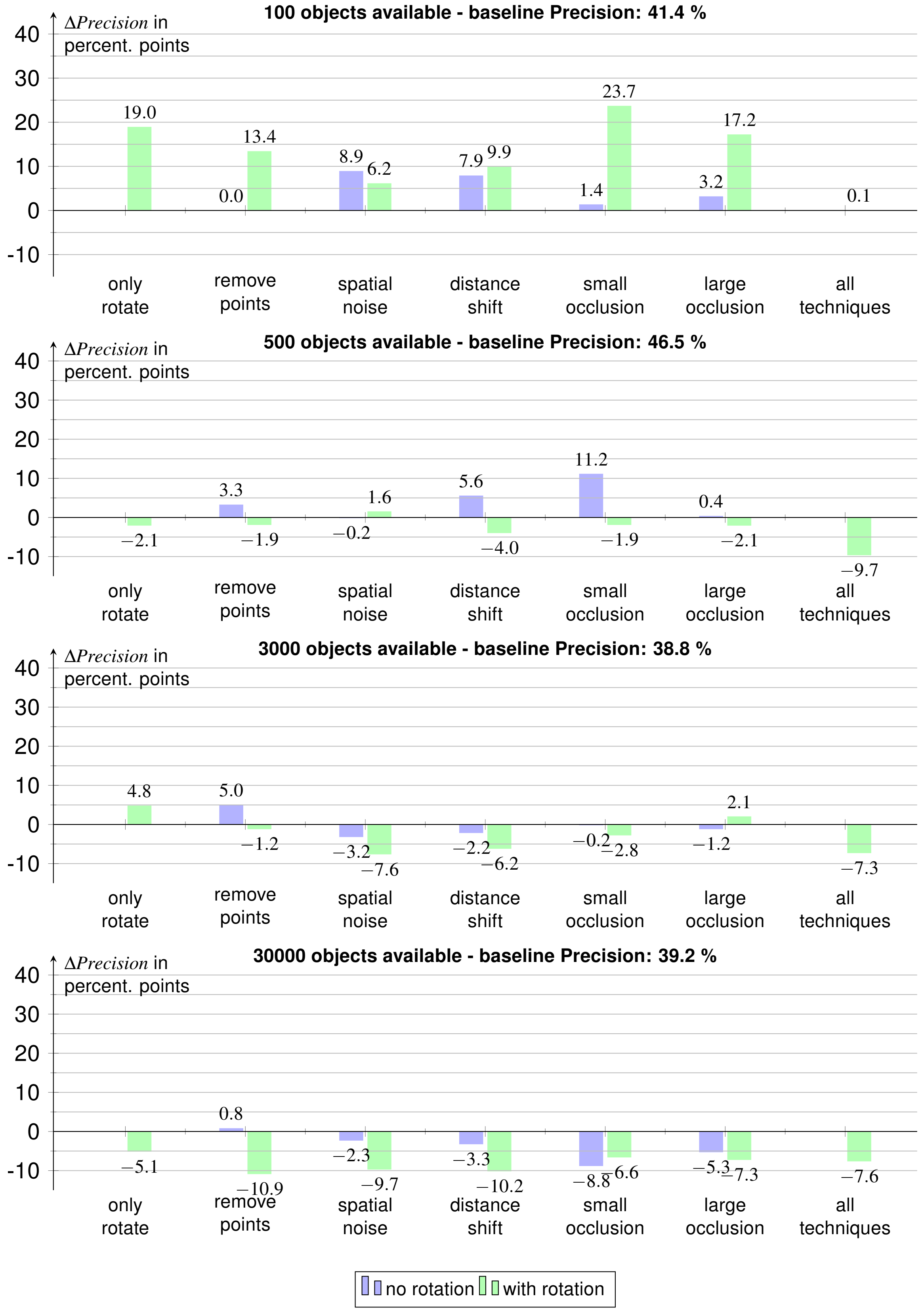


Fig. 6-4: Results of Experiment 2 regarding precision for varying the number of objects available for insertion.

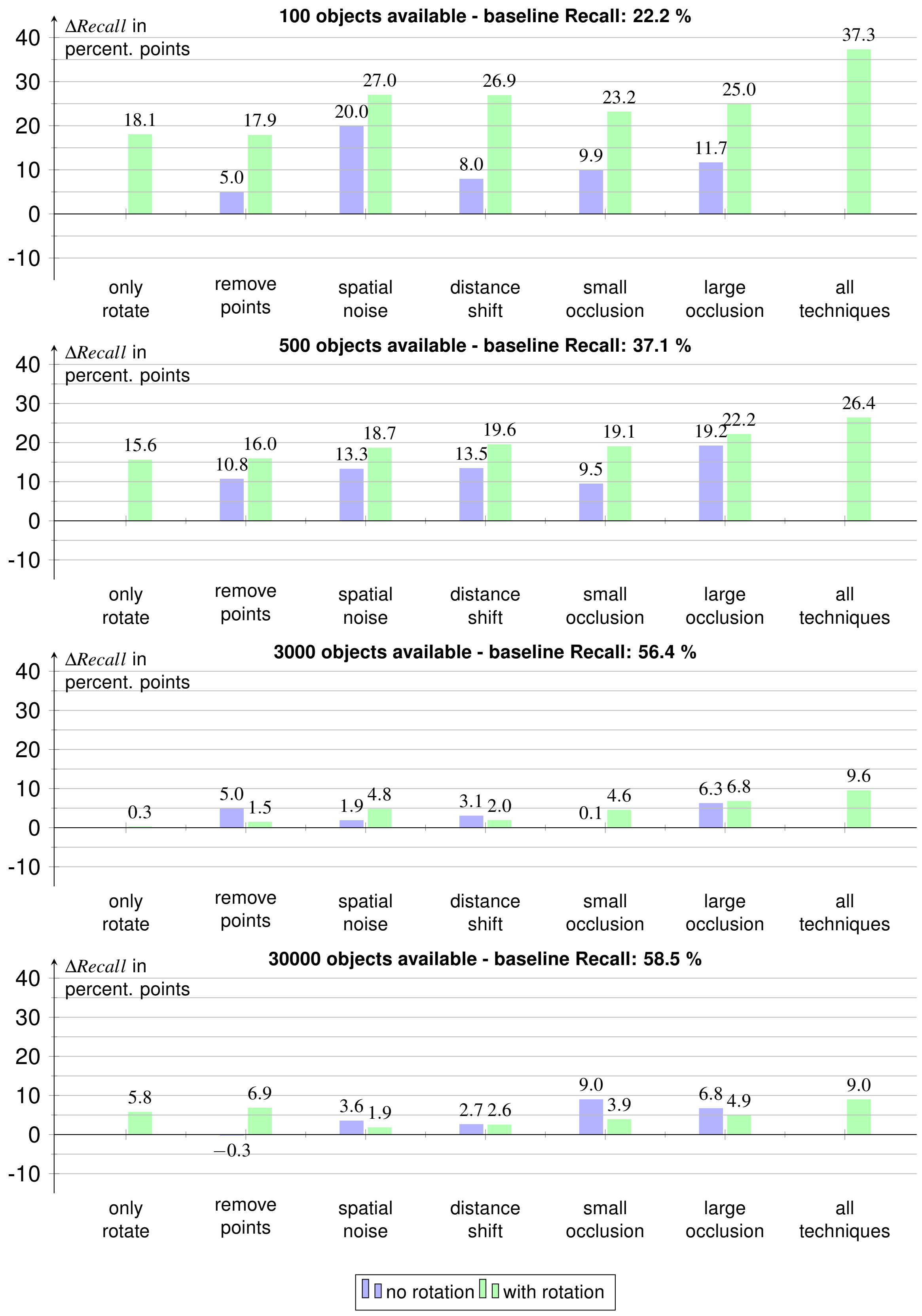


Fig. 6-5: Results of Experiment 2 regarding recall for varying the number of objects available for insertion.

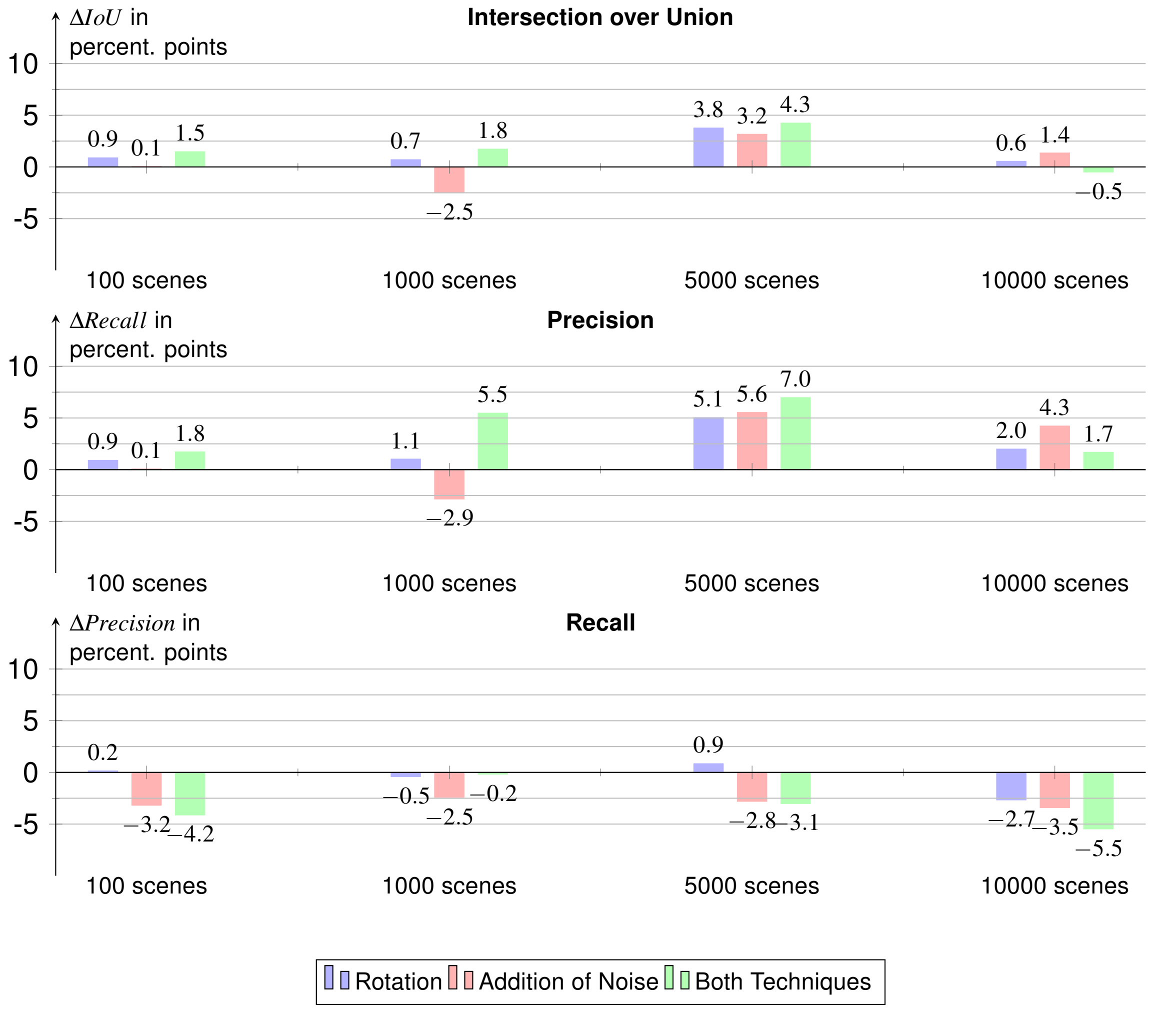


Fig. 6-6: Results of Experiment 2 for varying the number of empty scenes available. The baseline metrics can be found in figure 6-2

6.3 Experiment 3: Comparing Semi-Artificial and Real Data Sets

This experiment aims to compare semi-artificial samples to real data sets. As mentioned in section 5.3, due to the possibly high similarity between training and validation data, the results of this experiment is only able to provide hints and reveal tendencies.

Supplementing Manually Annotated Data

For the first comparison, two semi-artificial data sets are created. The first one is created by reinserting objects into the empty scenes obtained by splitting a manually annotated data set. The second one uses the same empty scenes, but consists of inserted objects extracted from the recordings on the test track. The performances of the CNNs trained on this data are visualized in figure 6-7.

The differences between the insertion of original objects compared to objects from the test track are rather small. IoU and Precision seem to increase when inserting objects from the test track.

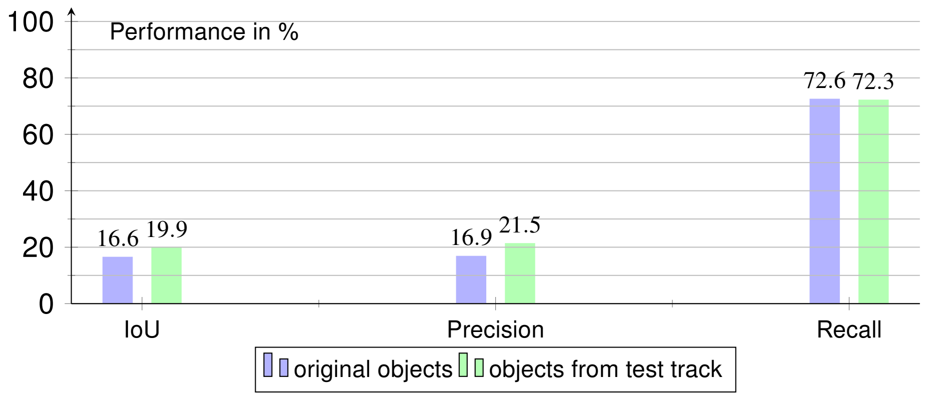


Fig. 6-7: Comparison of the performances of semi-artificial data sets, once with objects originating from the same scene and once from the test track.

However 3.3pp and 4.6pp respectively seem too low to attribute this change to objects used. It should be noted again that we focus on qualitative results in this experiment. The similarity in performances can indicate that semi-artificial data can indeed be used as supplement for real data. The similar performances suggest that the quality with respect to the training process is comparable. Therefore, it seems plausible that they can be used in the same training data set.

Substituting Manually Annotated Data

Figure 6-8 shows the performance metrics for each epoch of the training. The IoU of the CNN trained on the semi-artificial data set is mostly indistinguishable from the performance if trained on real data. When examining precision and recall, it can be seen that real samples show a higher precision while semi-artificial samples correlate with a higher recall.

The small differences in precision and recall and the almost identical IoU support the hypothesis that semi-artificial samples can serve as a substitution of real samples. A possible explanation for the differing precision and recall is that the objects inserted into the scene might show a higher intra-class variability than the original objects. As the original scenes are all from the same measurement, it is probable that the recorded scenes contain a few people in similar poses. If a pedestrian passes by the measurement vehicle, he or she can be found in multiple recorded scenes. However, a pedestrian passing by presumably does not change his posture very often, for example by changing from walking to running and back quickly. The pedestrians in the controlled environment however have been recorded in many different postures, as explained in section 4.3.1. As already argued in section 6.1, it is possible that a higher variety of objects leads to an increasing recall. It is also possible that the increase of the recall diminishes the precision. If the model is very sensitive about predicting an object, more points are classified as pedestrian, decreasing the number of false negatives but increasing the false positives. As explained in section 2.12, the lower false negatives increase the recall while the increasing false positives lead to a decreasing precision.

Regarding the results of the two tests performed in experiment 3, no differences between semi-artifical and real pointclouds can be identified. The data suggests that the two types of samples can be used equally, the hypothesis that semi-artificial samples can be used either as supplement or as substitution for real pointclouds is supported by the results.

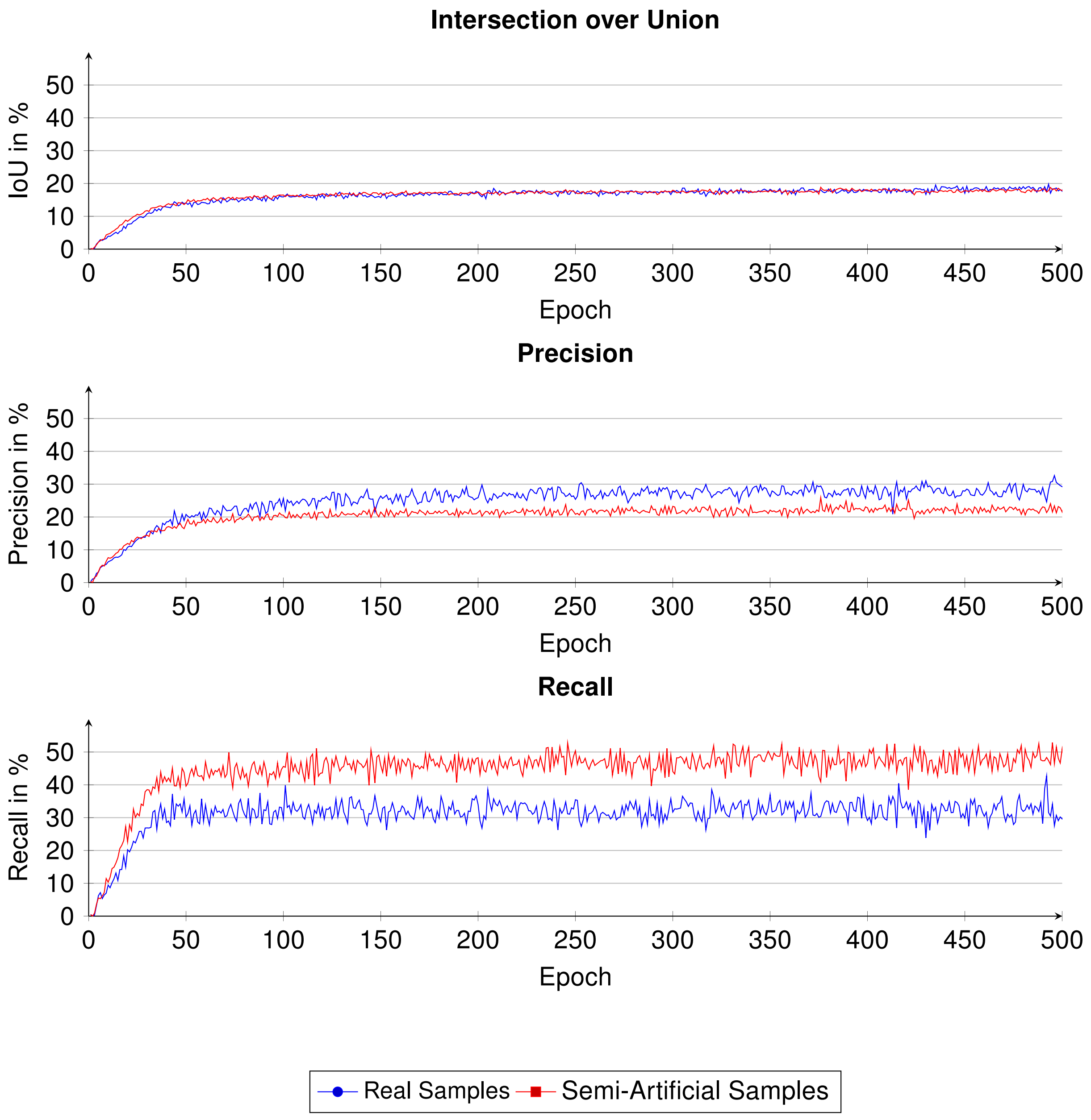


Fig. 6-8: Learning curve comparing a training on a real data set with a semi-artificial data set. Both data sets have pedestrians positioned at the same locations.

6.4 Experiment 4: Data Sets with Multiple Samples

As mentioned in section 5.4, this experiment examines if the creation of semi-artificial samples is not only possible for the semantic segmentation of one class but also for multiple classes simultaneously. While it is reasonable to focus on one class when evaluating the new developed augmentation techniques, semantic segmentation for one class only is not sufficient to create an environment model for automated driving. The results for a neural network trained on a semi-artificial data set containing pedestrians and vehicles is found in figure 6-9. A bar chart shows IoU, precision and recall sorted by class.

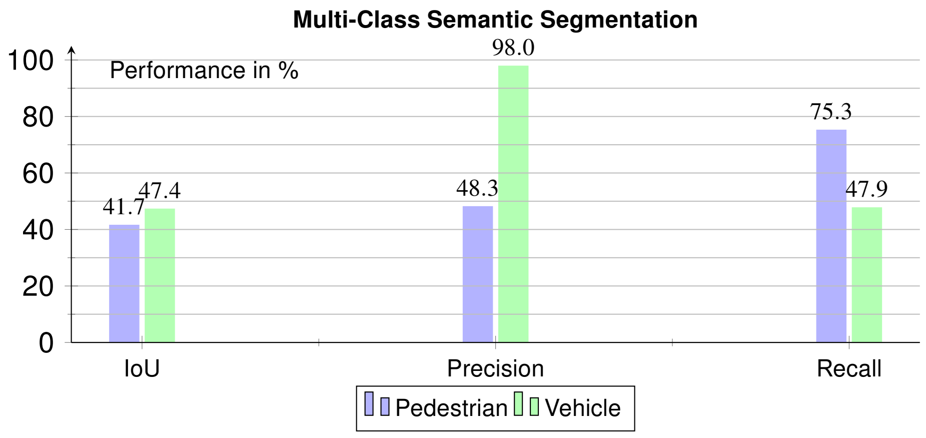


Fig. 6-9: Results of Experiment 4: Performances for the semantic-segmentation of pedestrians and vehicles simultaneously

The metrics regarding the class pedestrian are not diminished by introducing a higher number of classes. The CNN developed for this thesis seems to have a sufficient capacity for the semantic segmentation of multiple classes. Additionally it seems that the network is able to reach reasonable performance regarding the semantic segmentation of vehicles. However, due to the small validation data set, the high precision of 98% should not lead to the assumption that it is in general possible to achieve such high precision with the presented techniques to create semi-artificial data.

6.5 Discussion

In this section, the results obtained in the individual experiments are summarized and used to draw conclusions regarding the questions raised in chapter 3. Limitations of the obtained results are shown. Additionally, questions left out by the experiments or emerging from their results are presented.

Reciting chapter 3, the main research question this work aims to answer is: *Which augmentation techniques can be applied to LiDAR-pointclouds in order to increase the performance of a neural network that performs semantic segmentation on these pointclouds?* Answering the three secondary questions introduced might help finding an answer.

How does the performance of a CNN trained on semi-artificial samples compare to state-of-the-art performances of CNNs trained on manually labeled pointclouds from recorded traffic scenes?

The metrics achieved by the various trainings indicate that performances comparable to state of the art results can be achieved by training on semi-artificial data. The results furthermore suggest that reasonable performances can be achieved even when creating data sets where some of the empty scenes and inserted objects are used multiple times. The performances using 30000 available objects are close to the ones obtained by only using 3000 objects.

In which way is the performance of a CNN influenced by traditional augmentation techniques applied to its training pointclouds?

Studying the results of experiment 2 leads to the assumption that an increase regarding all three metrics is achieved for data sets where objects to insert are scarce. This conclusion however seems obvious. The observation, that an increasing recall is detected for all augmentation technique is more interesting. Especially considering VRU, a high recall is important for automated driving applications in order to prevent accidents.

To which extend do the developed augmentation techniques lead to a supplement or even substitution of manually annotated pointclouds from recorded traffic scenes?

The results obtained suggest that semi-artificial pointclouds are suitable both as supplement and as substitution of manually annotated data. Small differences in precision and recall can arise when comparing real pointclouds to semi-artificial ones. A possible explanation for this is that the objects extracted from the test track can be intentionally recorded with a high variability of different poses while objects in real recordings typically do not alternate their poses extensively from one moment to another.

Research Question

Regarding the results obtained in this work and putting them into context regarding the three secondary questions leads to the following conclusion:

The creation of semi-artificial samples turned out to be an effective augmentation technique. Data sets created using this technique showed high similarities in the performance metrics compared to a real data set. Traditional augmentation techniques are most beneficial to increase recall, which is an important metric in the context of automated driving. However, a decrease in precision can go along with this and should not be set aside.

Recommendations for Further Evaluation

The high performance variance determined especially in experiment 2 demand for a further examination of the traditional augmentation techniques. An experiment not done in this work is the evaluation of the influence of augmentation parameters. Additionally, new augmentation techniques can be developed and evaluated. For example, the objects could be distorted in height and width the whole semi-artificial samples could be flipped.

The augmentation techniques developed could be tested with another sensor. As described in section 4.3.1, the sensor used only consists of 16 layers. Sensors with a higher vertical resolution can be tested. The pointclouds obtained by such a sensor are denser and might therefore offer more room to tune the augmentation.

A weakness in this work is the rather small amount of manually annotated pointclouds for evaluation. It is recommended to repeat the experiments with a higher amount of evaluation data. However, this data is hard to obtain. An approach to bypass this problem could be the use of an another LiDAR-sensor. Few public data sets containing labeled LiDAR pointclouds exist, but choosing a sensor that is used in one of those public available data sets might enable the use of that data set for evaluation.

7 Conclusion & Outlook

In this thesis, a concept for the augmentation of LiDAR-pointclouds was developed and evaluated. The techniques presented were successfully used to train a CNN for semantic segmentation. Two types of augmentation were proposed: First, the augmentation techniques in a traditional sense, i.e. the application of label preserving transformations to a single pointcloud. Second, the creation of semi-artificial samples, where objects from one measurement are inserted into scenes of another one.

The biggest contribution of this work is both the provision of a software framework allowing the augmentation of LiDAR-pointclouds and the analysis conducted on top of this research, showing that semi-artificial pointclouds can be used as supplement or substitution of real recordings. This enables the creation of large pointcloud data sets without being restricted by the expensive manual labeling process.

Regarding the traditional augmentation techniques developed, no statement could be made about which augmentation techniques lead to a generally higher performance. Instead, the benefit of a technique seemed to be highly dependent on the data used for training and validation. This leads to the assumption that techniques have to be carefully selected and possibly tuned in order to profit from these augmentations. The influence of the augmentation parameters should be studied further.

Though IoU is one of the most popular metrics used to evaluate semantic segmentation, it turned out that precision and recall enable a deeper evaluation and understanding of the influence of augmentation techniques. For both types of augmentation techniques, recall showed the highest increase in performance. This has been attributed to the fact that augmenting the objects comes with a higher intra-class variability of the objects' class while influencing the background variability in a smaller amount.

An experiment performed for multi-class semantic segmentation hints that the results obtained for the semantic segmentation of pedestrians could be transferred to other classes. However, no in-depth evaluation has been done. Further research regarding the semantic segmentation of multiple classes is needed.

The CNN developed has successfully been implemented in the test vehicle of the IKA. The input transformation and semantic segmentation show a processing time of up to 9 ms for the current sensor setup. Clustering the segmented pointcloud for an environment model costs an additional 10 ms. This indicates the possibility for real time application considering that a typical LiDAR-sensor produces pointclouds with a frequency of 10-20 Hz.

A byproduct of this work is the suggestion and the development of a method to facilitate manual labeling of pointclouds. Simultaneously to the recording of LiDAR-pointclouds, drone images can be used to receive a top view of the measured scenes. For a human annotator, labeling these images is presumably easier than labeling pointclouds. By overlaying the images and pointclouds, it is possible to use the images' labels for the automatic annotation of the corresponding pointclouds. For further acceleration of the labeling process, a neural network can be used to obtain a prior suggestion for labels.

Finally the author of this work would like to suggest three ideas that emerged during the development, seeming particularly interesting for further research:

As presented in chapter 2, [CUB18] claims a neural network that is able to learn the augmentation techniques which probably lead to the highest increase in performance. Transferring the concept from the image domain to pointclouds could help identifying and tuning the most suitable techniques developed.

Furthermore, the pointcloud representation used for evaluation did not consider intensity measurements. As described in chapter 4, this opens up the possibility to use simulated data. Simulations, real data and semi-artificial data can be mixed or semi-artificial samples can be created using real empty scenes and simulated objects or vice versa.

In this work, semi-artificial samples are created by inserting objects that shall be classified into empty scenes. The insertion of static objects that do not have a distinct class seems like an interesting idea. Unsupervised learning, e.g. clustering, could be used to identify objects in empty scenes. With the techniques developed, these objects could be inserted into other scenes without the need to know their class. This way, semi-artificial scenes can be created with a higher variability.

References

[BAD17] V. BADRINARAYANAN, A. KENDALL and R. CIPOLLA
SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (Dec. 1, 2017), pp. 2481–2495

[BEL18] J. BELTRAN et al.
BirdNet: A 3D Object Detection Framework from LiDAR Information, *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI: IEEE, Nov. 2018, pp. 3517–3523

[BEY18] M. Y. BEYEN
Erzeugung einer Laserscanner-basierten Hinderniskarte und Extraktion von Befahrbarkeitsgrenzen für die lokale Trajektorienoptimierung, Master's Thesis, Aachen, Germany: RWTH Aachen University, Sept. 2018

[BLO17] M. BLOICE, C. STOCKER and A. HOLZINGER
Augmentor: An Image Augmentation Library for Machine Learning, *The Journal of Open Source Software* 2.19 (Nov. 4, 2017), p. 432

[BOY14] A. BOYKO and T. FUNKHOUSER
Cheaper by the dozen: group annotation of 3D data, *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*, the 27th annual ACM symposium, Honolulu, Hawaii, USA: ACM Press, 2014, pp. 33–42

[BUS18] A. BUSLAEV et al.
Albumentations: fast and flexible image augmentations, *arXiv:1809.06839 [cs]* (Sept. 18, 2018), arXiv: `1809.06839`

[CHA02] N. V. CHAWLA et al.
SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research* 16 (June 1, 2002), pp. 321–357

[CHA17] R. Q. CHARLES et al.
PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, July 2017, pp. 77–85

[COM02] D. COMANICIU, D. COMANICIU and P. MEER
Mean shift: A robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), pp. 603–619

[CON19] O. S. M. CONTRIBUTORS
Planet dump retrieved from https://planet.osm.org, 2019, URL: `https://www.openstreetmap.org`

[COR16] M. CORDTS et al.
The Cityscapes Dataset for Semantic Urban Scene Understanding, 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), Las Vegas, NV, USA: IEEE Computer Society, 2016, pp. 3213–3223

[CUB18] E. D. CUBUK et al.
AutoAugment: Learning Augmentation Policies from Data, *arXiv:1805.09501 [cs, stat]* (May 24, 2018), arXiv: `1805.09501`

[DEN09] J. DENG et al.
ImageNet: A large-scale hierarchical image database, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL: IEEE, June 2009, pp. 248–255

[ESS09] A. ESS et al.
Segmentation-Based Urban Traffic Scene Understanding, *Procedings of the British Machine Vision Conference 2009*, British Machine Vision Conference 2009, London: British Machine Vision Association, 2009, pp. 84.1–84.11

[EVA04] M. EVANS and J. ROSENTHAL
Probability & Statistics - The Science of Uncertainty, New York: W.H. Freeman and Company, 2004

[GAR17] A. GARCIA-GARCIA et al.
A Review on Deep Learning Techniques Applied to Semantic Segmentation, *CoRR* abs/1704.06857 (2017), arXiv: `1704.06857`

[GEI12] A. GEIGER, P. LENZ and R. URTASUN
Are we ready for autonomous driving? The KITTI vision benchmark suite, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, kitti_benchmark, Providence, RI: IEEE, June 2012, pp. 3354–3361

[GEI13] A. GEIGER et al.
Vision meets robotics: The KITTI dataset, *The International Journal of Robotics Research* 32.11 (Sept. 2013), kitti_data, pp. 1231–1237

[GOO16] I. GOODFELLOW, Y. BENGIO and A. COURVILLE
Deep learning, Adaptive computation and machine learning, Cambridge, Massachusetts: The MIT Press, 2016, 775 pp.

[GRO18] P.-N. GRONERTH, B. HAHN and L. ECKSTEIN
Automated Data Generation for Training of Neural Networks by Recombining Previously Labeled Images, *Advanced Microsystems for Automotive Applications 2017*, ed. by C. ZACHÄUS, B. MÜLLER and G. MEYER, Lecture Notes in Mobility, Springer International Publishing, 2018, pp. 125–135

[HE16] K. HE et al.
Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, June 2016, pp. 770–778

[IAN16] F. N. IANDOLA et al.
SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, *arXiv:1602.07360 [cs]* (Feb. 23, 2016), squeezenet

[JAR18] M. JARITZ et al.
Sparse and Dense Data with CNNs: Depth Completion and Semantic Segmentation, *arXiv:1808.00769 [cs]* (Aug. 2, 2018), arXiv: `1808.00769`

[JI13] S. JI et al.
3D Convolutional Neural Networks for Human Action Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (Jan. 2013), pp. 221–231

[KOP11] R. KOPPER, F. BACIM and D. A. BOWMAN
Rapid and accurate 3D selection by progressive refinement, *2011 IEEE Symposium on 3D User Interfaces (3DUI)*, progessive_refinement, Singapore, Singapore: IEEE, Mar. 2011, pp. 67–74

[KRI12] A. KRIZHEVSKY, I. SUTSKEVER and G. E. HINTON
ImageNet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems 25* 60.6 (2012), pp. 84–90

[LAM18] B. LAMPE, L. ECKSTEIN and P. GRONERTH
Erstellung von automatisch gelabelten, semi-künstlichen und künstlichen Daten, *Automobiltechnische Zeitschrift (ATZ)* 23.4 (Aug. 2018), pp. 26–29

[LEC15] Y. LECUN, Y. BENGIO and G. HINTON
Deep learning, *Nature* 521.7553 (May 2015), pp. 436–444

[LEC89] Y. LECUN et al.
Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation* 1.4 (Dec. 1989), pp. 541–551

[LEM17] J. LEMLEY, S. BAZRAFKAN and P. CORCORAN
Smart Augmentation - Learning an Optimal Data Augmentation Strategy, *IEEE Access* 5 (2017), pp. 5858–5869, arXiv: `1703.08383`

[LI 12] LI DENG
The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web], *IEEE Signal Processing Magazine* 29.6 (Nov. 2012), pp. 141–142

[LI16] B. LI, T. ZHANG and T. XIA
Vehicle Detection from 3D Lidar Using Fully Convolutional Network, *Robotics: Science and Systems XII*, Robotics: Science and Systems 2016, Robotics: Science and Systems Foundation, 2016

[MA18] F. MA and S. KARAMAN
Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image, *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD: IEEE, May 2018, pp. 1–8

[MAC67] J. MACQUEEN
Some methods for classification and analysis of multivariate observations, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, The Regents of the University of California, 1967

[MEH18] P. MEHTA et al.
A high-bias, low-variance introduction to Machine Learning for physicists (Mar. 23, 2018), introduction_ml

[MIT97] T. M. MITCHELL
Machine learning, International ed., [Reprint.], McGraw-Hill series in computer science, OCLC: 846511832, New York, NY: McGraw-Hill, 1997, 414 pp.

[OSB05] L. OSBORNE et al.
Clarus Concept of Operations, FHWA-JPO-05-072, U.S. Department of Transportation, Oct. 2005

[OSH15] K. O'SHEA and R. NASH
An Introduction to Convolutional Neural Networks, *arXiv:1511.08458 [cs]* (Nov. 26, 2015), arXiv: `1511.08458`

[PAR18] W. C. PARTNERS and Y. DÉVELOPPEMENT
Automotive LiDAR Market Report - A report on automotive imaging from Woodside Capital Partners and Yole Développement, Apr. 2018

[POH17] T. POHLEN et al.
Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, July 2017, pp. 3309–3318

[POL19] N. G. POLSON and V. SOKOLOV
Bayesian Regularization: From Tikhonov to Horseshoe, *arXiv:1902.06269 [stat]* (Feb. 17, 2019), arXiv: `1902.06269`

[RIE99] M. RIESENHUBER and T. POGGIO
Hierarchical models of object recognition in cortex, *Nature Neuroscience* 2.11 (Nov. 1999), p. 1019

[ROS58] F. ROSENBLATT
The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65.6 (1958), pp. 386–408

[RUM86] D. E. RUMELHART, G. E. HINTON and R. J. WILLIAMS
Learning representations by back-propagating errors, *Nature* 323.6088 (Oct. 1986), p. 533

[RUS15] O. RUSSAKOVSKY et al.
ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision* 115.3 (Dec. 1, 2015), pp. 211–252

[SHA12] T. SHAO et al.
An interactive approach to semantic modeling of indoor scenes with an RGBD camera, *ACM Transactions on Graphics* 31.6 (Nov. 1, 2012), indoorlabeltool, p. 1

[SHL18] S. E. SHLADOVER
Connected and automated vehicle systems: Introduction and overview, *Journal of Intelligent Transportation Systems* 22.3 (May 4, 2018), pp. 190–200

[SIM03] P. SIMARD, D. STEINKRAUS and J. PLATT
Best practices for convolutional neural networks applied to visual document analysis, *Seventh International Conference on Document Analysis and Recognition*, vol. 1, Edinburgh, UK: IEEE Comput. Soc, 2003, pp. 958–963

[SIM14] K. SIMONYAN and A. ZISSERMAN
Very Deep Convolutional Networks for Large-Scale Image Recognition, *arXiv:1409.1556* (Sept. 4, 2014), arXiv: `1409.1556`

[SIN18] K. K. SINGH et al.
Hide-and-Seek: A Data Augmentation Technique for Weakly-Supervised Localization and Beyond, *arXiv:1811.02545 [cs]* (Nov. 6, 2018), arXiv: `1811.02545`

[SRI14] N. SRIVASTAVA et al.
Dropout: a simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research* 15.1 (Jan. 1, 2014), pp. 1929–1958

[STA18] STATISTISCHES-BUNDESAMT
Unfallentwicklung auf Deutschen Strassen 2017 - Begleitmaterial zur Pressekonferenz am 12. Juli 2018 in Berlin, 2018

[SUT98] R. S. SUTTON and A. G. BARTO
Reinforcement learning: an introduction, Adaptive computation and machine learning, Cambridge, Mass: MIT Press, 1998, 322 pp.

[SZE15] C. SZEGEDY et al.
Going deeper with convolutions, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA: IEEE, June 2015, pp. 1–9

[THO16] M. THOMA
A Survey of Semantic Segmentation, *arXiv:1602.06541 [cs]* (Feb. 21, 2016), arXiv: `1602.06541`

[VEI14] M. VEIT and A. CAPOBIANCO
Go'Then'Tag: A 3-D point cloud annotation technique, *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, gothentag, Minneapolis, MN: IEEE, Mar. 2014, pp. 193–194

[VEL07] VELODYNE
Velodyne's HDL-64E: A High Definition LiDAR Sensor for 3D Applications, Whitepaper, Oct. 2007

[WEN92] J. WENG, N. AHUJA and T. HUANG
Cresceptron: a self-organizing neural network which grows adaptively, *IJCNN International Joint Conference on Neural Networks*, vol. 1, Baltimore, MD, USA: IEEE, 1992, pp. 576–581

[WON15] Y.-S. WONG, H.-K. CHU and N. J. MITRA
SmartAnnotator An Interactive Tool for Annotating Indoor RGBD Images, *Computer Graphics Forum* 34.2 (May 2015), smartannotator, pp. 447–457

[WU17] B. WU et al.
SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud, *arXiv:1710.07368 [cs]* (Oct. 19, 2017), squeezeseg

[YUE18] X. YUE et al.
A LiDAR Point Cloud Generator: from a Virtual World to Autonomous Driving, *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval* (2018), arXiv: `1804.00103`

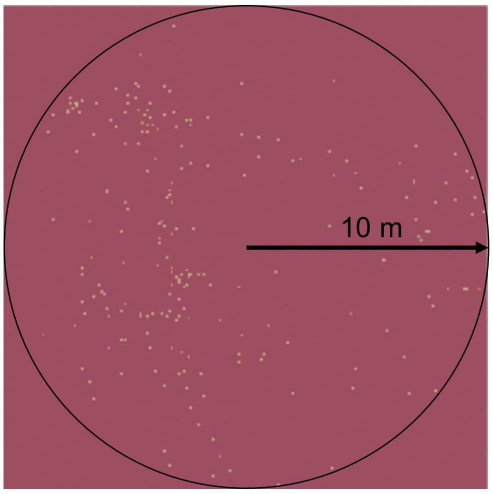
[ZHA18] Y. ZHANG and T. FUNKHOUSER
Deep Depth Completion of a Single RGB-D Image, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA: IEEE, June 2018, pp. 175–185

[ZHO18] Y. ZHOU and O. TUZEL
VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA: IEEE, June 2018, pp. 4490–4499

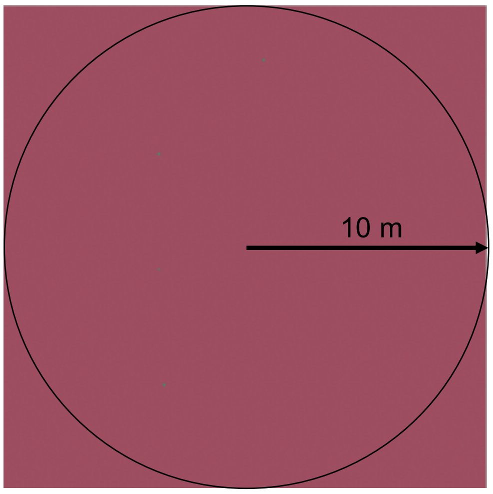
A Appendix

Sensor	- 16 channels - Measurement Range: max. 100 m - Range Accuracy typically up to 3 cm - Vertical Field of View: -15.0° to +15.0° - Vertical Angular Resolution: 2° - Horizontal Field of View: 360° - Horizontal Angular Resolution: 0.1° - 0.4° - Rotation Rate: 5 Hz - 20 Hz
Laser	- Wavelength: 903 nm
Output	- Single Return Mode: max. 300.000 points per second - Dual Return Mode: max. 600.000 points per second - UDP-Packets: Time of Flight, Reflectivity, Rotation Angle, Timestamp

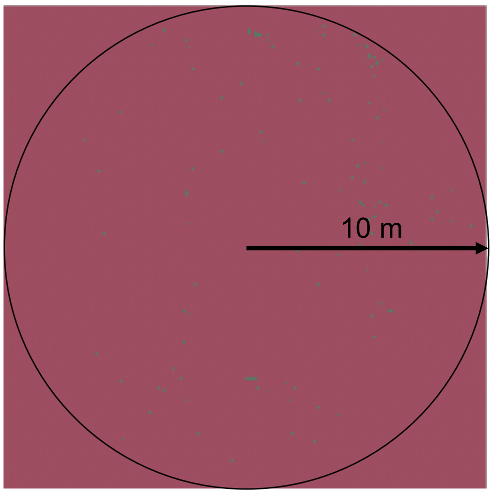
Tab. A-1: Velodyne Puck VLP-16 specifications. Information taken from https://velodynelidar.com/



a) Locations of pedestrians.



b) Locations of bicyclists (4 objects)



c) Locations of vehicles

Fig. A-1: Locations of objects in the evaluation data set. a) Pedestrians, b) Bicyclists c) Vehicles

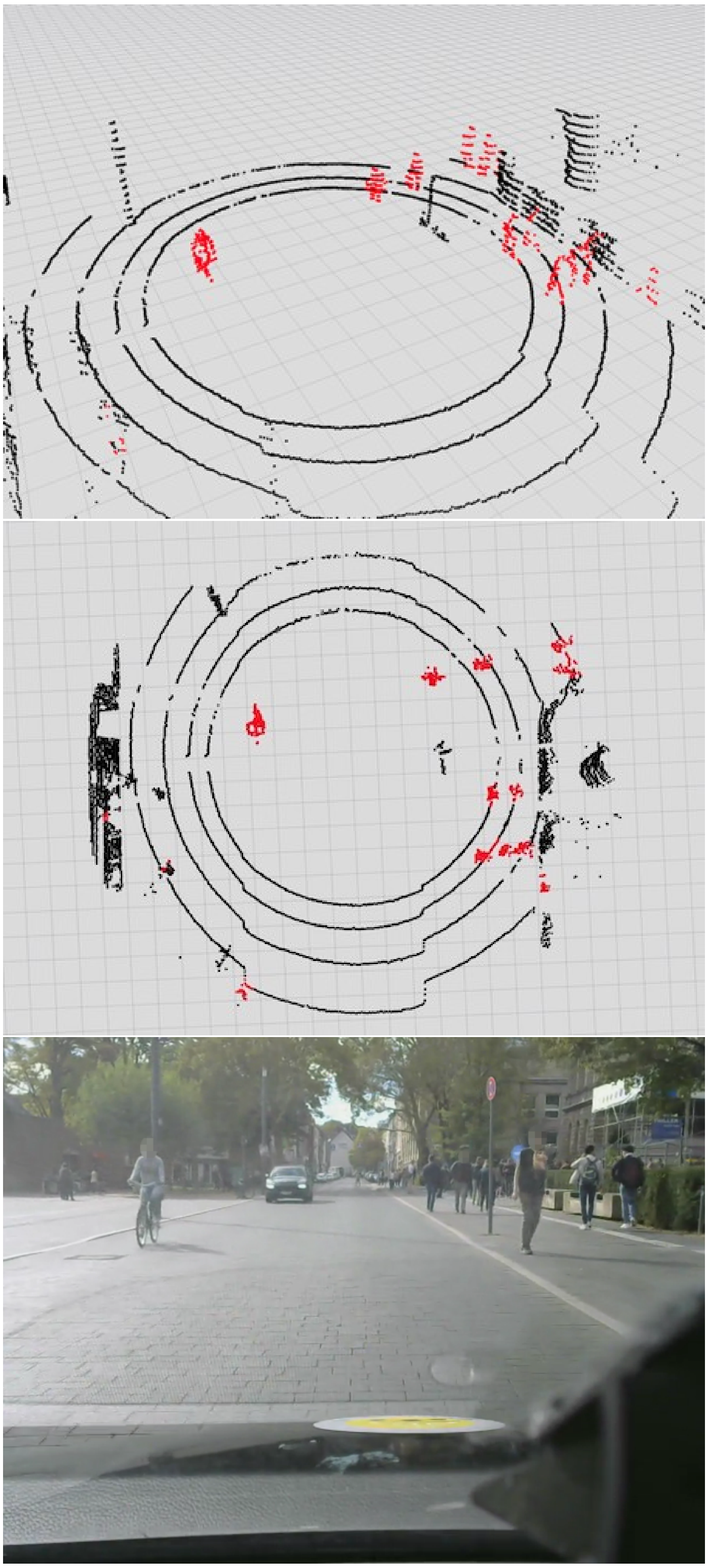
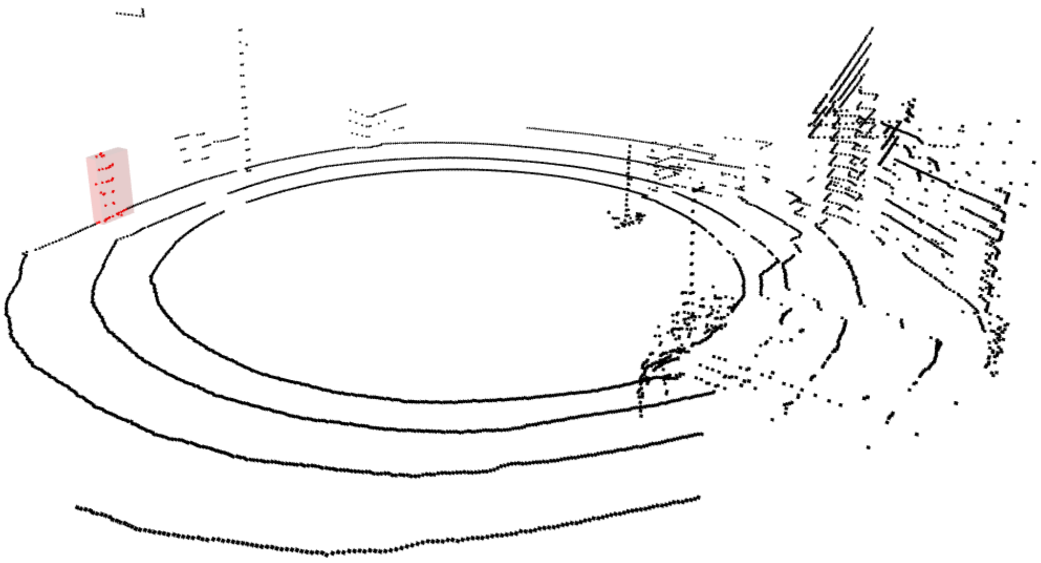


Fig. A-2: Qualitative results obtained using a matrix-representation of size 100x176x16 with a maximum distance of 10 m.


a) Good result

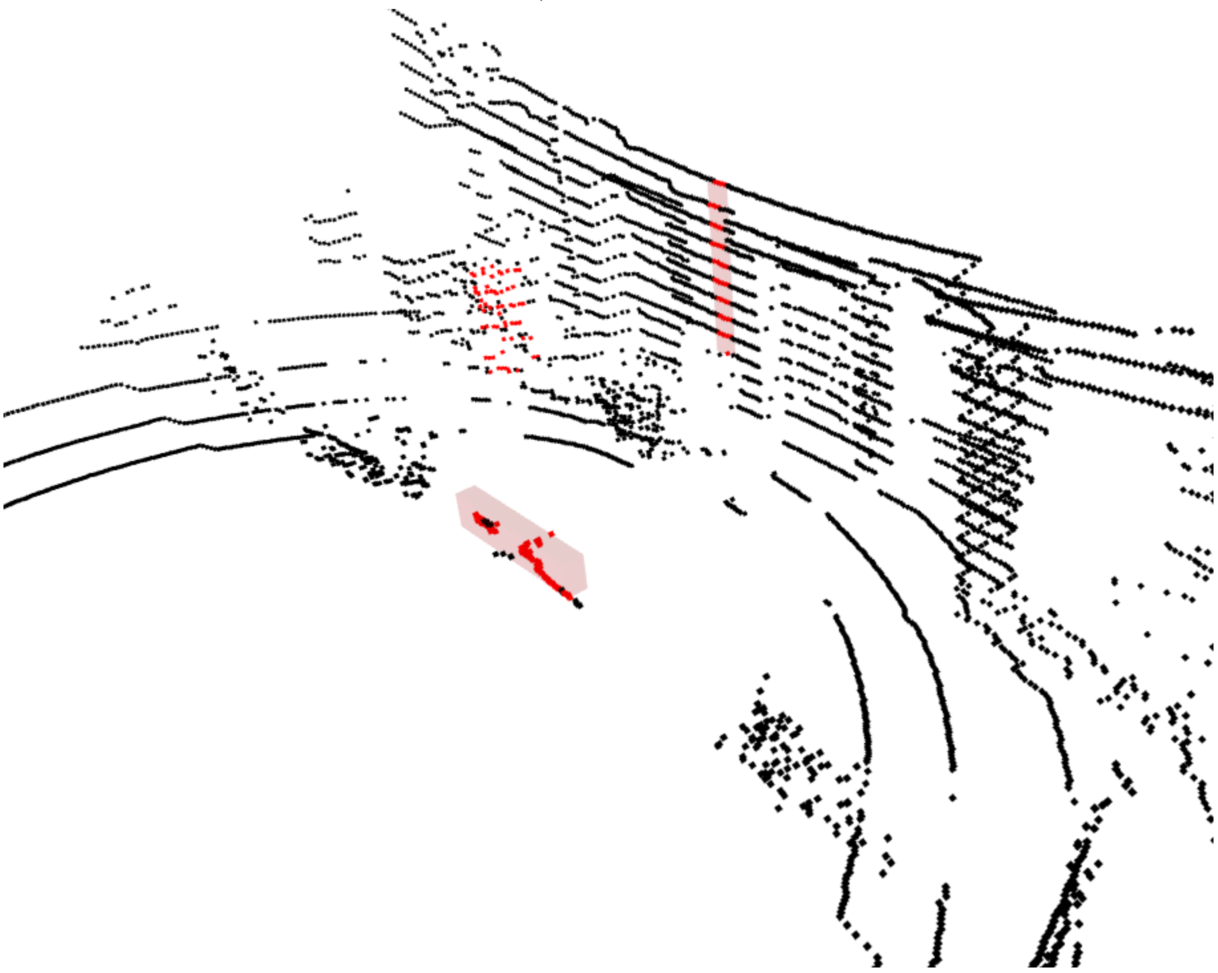

b) Result showing false positives

Fig. A-3: Example for a qualitative comparison of two predictions. a) good results b) false positives.

100 objects	no augm.	rotate	remove	+ rot	noise	+ rot	shift	+ rot	small occ.	+ rot	large occ.	+ rot	all
Mean IoU	17,91	13,30	1,42	12,07	11,11	13,71	6,00	15,27	5,20	18,61	5,79	17,43	14,32
Mean Precis	41,36	18,95	0,01	13,44	8,94	6,16	7,93	9,90	8,51	23,72	3,34	17,24	0,08
Mean Recal	22,21	18,08	5,03	17,91	19,97	27,03	7,97	26,94	9,91	23,21	11,66	25,01	37,33
IoU 1	18,04	32,49	21,97	28,39	33,46	30,12	26,60	36,59	25,71	39,46	26,57	38,68	32,83
Precision 1	25,98	55,00	55,08	54,15	52,17	41,62	57,63	53,80	52,68	74,23	55,24	67,42	40,20
Recall 1	21,48	44,26	26,77	37,39	48,27	52,15	33,15	53,35	33,42	45,73	33,87	47,58	64,16
IoU 2	17,44	30,94	17,89	29,81	22,10	30,75	19,33	29,70	21,12	39,14	19,90	34,52	32,29
Precision 2	58,48	53,02	32,49	48,85	32,58	45,19	32,59	42,05	32,80	67,42	33,87	55,60	43,99
Recall 2	19,90	42,62	28,48	43,33	40,73	49,04	25,57	50,28	37,24	48,27	32,55	47,66	54,84
IoU 3	18,24	30,19	18,13	31,74	31,49	33,97	25,80	33,25	22,48	30,95	24,61	32,80	31,55
Precision 3	39,61	72,89	36,52	61,40	66,15	55,74	57,64	57,93	64,11	53,57	44,98	52,76	40,12
Recall 3	25,26	34,01	26,47	39,65	37,54	46,53	31,84	43,84	25,72	42,28	35,20	46,43	59,63

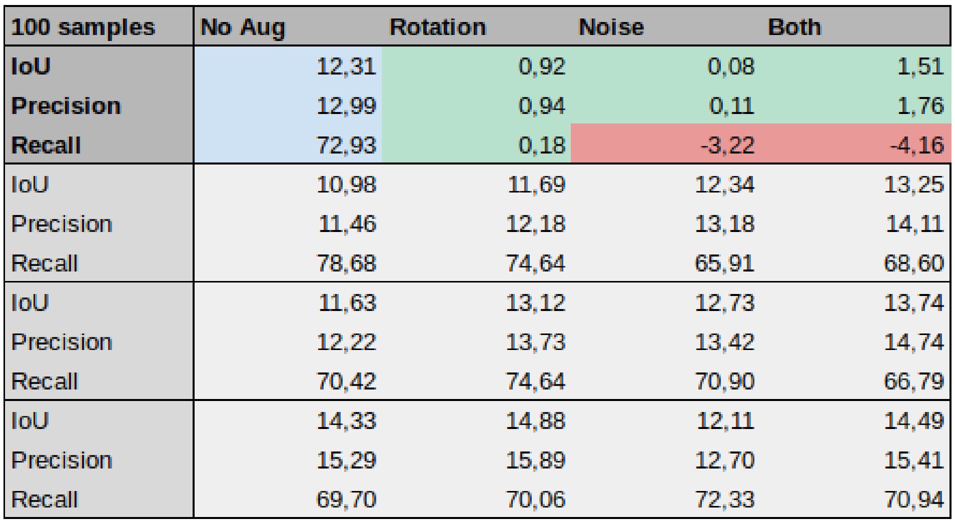
500 objects	nothing	rotate	remove	remove + rot	noise	noise + rot	shift	shift + rot	small occ.	small occ. + rot	large occ.	large occ. + rot	all
IoU	26,92	4,86	4,89	4,92	4,85	7,51	6,69	5,02	7,72	5,82	7,51	7,17	3,50
Precision	46,51	-2,08	3,29	-1,91	-0,17	1,56	5,60	-3,99	11,17	-1,91	0,40	-2,09	-9,66
Recall	37,13	15,64	10,76	15,97	13,31	18,71	13,48	19,58	9,52	19,06	19,24	22,20	26,43
IoU 1	28,99	29,23	33,75	30,35	34,75	31,38	36,49	29,69	38,47	27,43	36,85	37,08	33,99
Precision 1	53,68	41,94	58,95	39,61	52,93	45,80	67,10	37,40	65,24	34,05	50,37	48,26	41,24
Recall 1	38,66	49,10	44,11	56,51	50,28	51,70	44,45	59,02	48,39	58,50	57,84	61,54	65,92
IoU 2	24,44	36,01	30,09	33,39	30,25	39,16	30,71	34,49	34,85	34,00	32,08	34,52	28,65
Precision 2	41,01	50,98	47,58	47,09	42,92	56,32	41,49	48,42	60,71	46,61	43,68	43,84	35,57
Recall 2	37,64	55,07	45,01	53,45	50,60	56,25	54,17	54,52	44,95	55,68	54,72	61,89	59,56
IoU 3	27,32	30,09	31,59	31,76	30,31	32,74	33,62	31,63	30,59	36,77	34,36	30,66	28,61
Precision 3	44,85	40,39	42,88	47,12	43,17	42,10	47,74	41,76	47,09	53,15	46,70	41,18	33,76
Recall 3	35,09	54,13	54,55	49,35	50,43	59,56	53,20	56,60	46,61	54,40	56,54	54,55	65,21

Fig. A-4: Detailed results for data sets with 100 and 500 available objects.

3000 object	nothing	rotate	remove	remove + rot	noise	noise + rot	shift	shift + rot	small occ.	small occ. + rot	large occ.	large occ. + rot	all
IoU	29,56	2,77	4,83	-0,11	-0,97	-3,75	-0,28	-3,16	0,20	-0,42	1,09	2,83	-2,62
Precision	38,79	4,83	5,04	-1,20	-3,21	-7,64	-2,18	-6,21	-0,21	-2,79	-1,20	2,07	-7,32
Recall	56,39	0,29	4,97	1,52	1,92	4,83	3,08	1,95	0,11	4,58	6,30	6,83	9,56
IoU 1	30,60	34,70	37,15	26,07	30,73	29,37	28,85	22,69	32,35	31,27	32,00	31,58	28,58
Precision 1	40,38	50,14	47,98	31,89	38,47	36,32	34,97	27,26	42,87	39,70	40,55	39,84	34,66
Recall 1	55,82	52,98	62,21	58,83	60,42	63,24	62,21	57,49	56,88	59,54	60,30	66,70	61,89
IoU 2	29,06	33,91	29,99	33,30	27,71	22,34	32,33	30,31	31,52	26,11	31,16	38,60	27,83
Precision 2	38,24	45,92	38,77	43,20	35,20	26,39	41,31	38,06	40,29	30,71	38,04	49,76	32,48
Recall 2	57,76	56,45	56,98	59,25	53,43	59,27	59,79	59,83	59,16	63,55	63,26	63,26	66,03
IoU 3	29,01	28,38	36,03	28,98	27,31	25,72	26,65	26,20	25,39	30,02	28,77	26,98	24,41
Precision 3	37,76	34,80	44,75	37,68	33,07	30,74	33,55	32,42	32,59	37,59	34,18	32,99	27,27
Recall 3	55,59	60,61	64,90	55,64	61,07	61,15	56,41	57,70	53,47	59,83	64,50	59,69	69,93

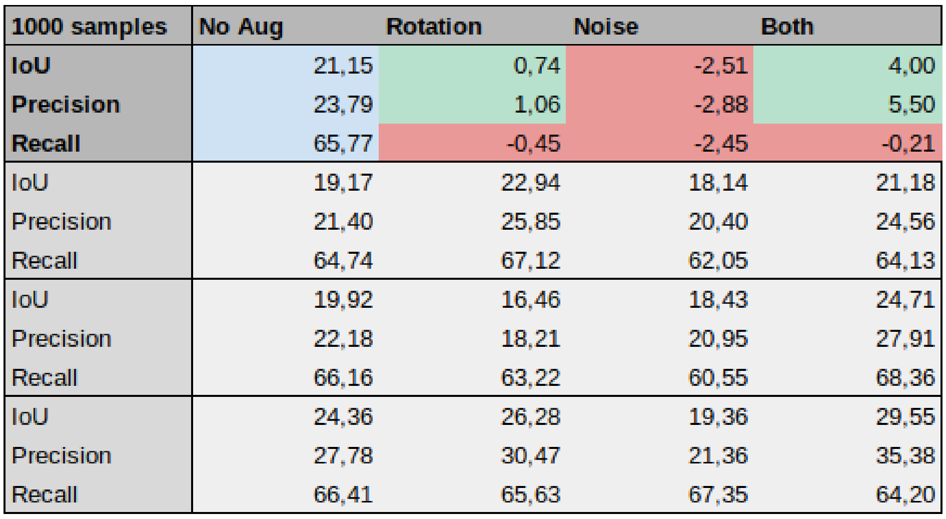
all objects	nothing	rotate	remove	remove + rot	noise	noise + rot	shift	shift + rot	small occ.	small occ. + rot	large occ.	large occ. + rot	all
IoU	30,71	-2,86	0,35	-6,11	-0,63	-6,21	-1,60	-6,23	-4,35	-3,68	-2,09	-3,78	-3,34
Precision	39,19	-5,13	0,84	-10,89	-2,33	-9,72	-3,26	-10,18	-8,83	-6,63	-5,30	-7,27	-7,64
Recall	58,45	5,80	-0,26	6,87	3,57	1,86	2,67	2,55	9,02	3,93	6,75	4,94	8,99
IoU 1	34,06	27,90	32,70	26,53	32,16	22,81	29,52	24,58	28,45	24,82	32,20	26,24	25,46
Precision 1	42,90	35,66	41,20	30,68	40,48	25,71	35,47	28,58	33,40	28,47	39,10	30,26	29,09
Recall 1	62,31	65,71	61,31	66,26	61,03	66,91	63,76	63,72	65,74	65,99	64,60	66,41	67,12
IoU 2	26,60	25,99	29,69	21,84	30,63	24,52	31,59	21,35	21,57	30,01	22,74	29,47	26,95
Precision 2	34,37	30,16	40,12	24,95	37,53	30,19	40,91	25,51	23,85	38,00	26,07	35,18	31,33
Recall 2	54,05	65,26	53,31	63,63	61,39	56,62	58,09	56,68	69,28	58,79	63,97	64,49	65,86
IoU 3	31,48	29,67	30,80	25,45	27,46	26,18	26,24	27,52	29,07	26,26	30,93	25,09	29,72
Precision 3	40,29	36,34	38,77	29,27	32,57	32,49	31,39	32,93	33,82	31,20	36,48	30,31	34,22
Recall 3	59,00	61,79	59,96	66,08	63,64	57,41	61,52	62,61	67,39	62,38	67,04	59,27	69,35

Fig. A-5: Detailed results for data sets with 3000 and 30000 available objects.



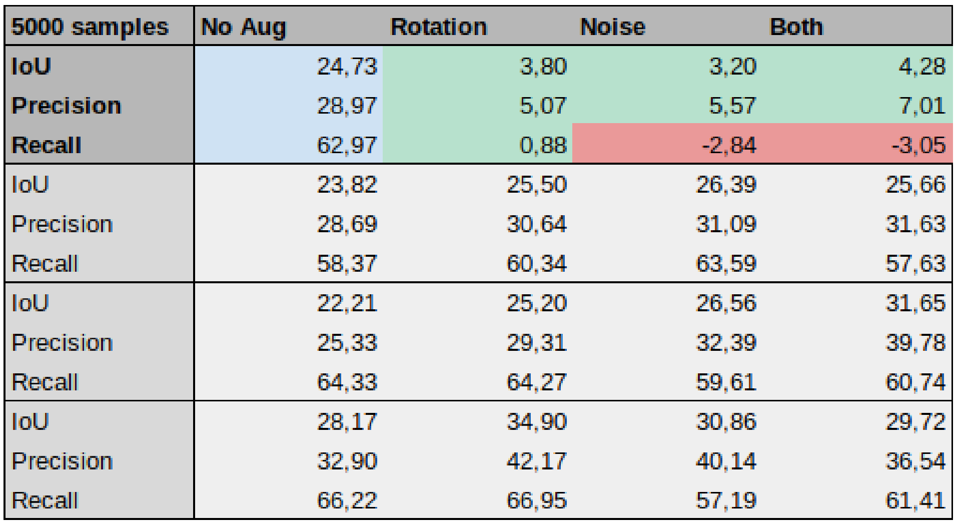
100 samples	No Aug	Rotation	Noise	Both
IoU	12,31	0,92	0,08	1,51
Precision	12,99	0,94	0,11	1,76
Recall	72,93	0,18	-3,22	-4,16
IoU	10,98	11,69	12,34	13,25
Precision	11,46	12,18	13,18	14,11
Recall	78,68	74,64	65,91	68,60
IoU	11,63	13,12	12,73	13,74
Precision	12,22	13,73	13,42	14,74
Recall	70,42	74,64	70,90	66,79
IoU	14,33	14,88	12,11	14,49
Precision	15,29	15,89	12,70	15,41
Recall	69,70	70,06	72,33	70,94

Fig. A-6: Detailed results for data sets 100 empty scenes.



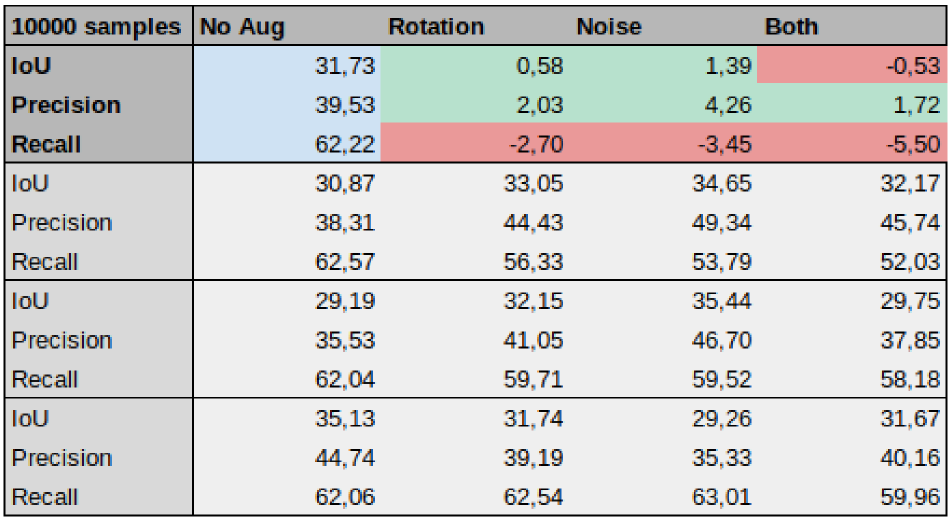
1000 samples	No Aug	Rotation	Noise	Both
IoU	21,15	0,74	-2,51	4,00
Precision	23,79	1,06	-2,88	5,50
Recall	65,77	-0,45	-2,45	-0,21
IoU	19,17	22,94	18,14	21,18
Precision	21,40	25,85	20,40	24,56
Recall	64,74	67,12	62,05	64,13
IoU	19,92	16,46	18,43	24,71
Precision	22,18	18,21	20,95	27,91
Recall	66,16	63,22	60,55	68,36
IoU	24,36	26,28	19,36	29,55
Precision	27,78	30,47	21,36	35,38
Recall	66,41	65,63	67,35	64,20

Fig. A-7: Detailed results for data sets 1000 empty scenes.



5000 samples	No Aug	Rotation	Noise	Both
IoU	24,73	3,80	3,20	4,28
Precision	28,97	5,07	5,57	7,01
Recall	62,97	0,88	-2,84	-3,05
IoU	23,82	25,50	26,39	25,66
Precision	28,69	30,64	31,09	31,63
Recall	58,37	60,34	63,59	57,63
IoU	22,21	25,20	26,56	31,65
Precision	25,33	29,31	32,39	39,78
Recall	64,33	64,27	59,61	60,74
IoU	28,17	34,90	30,86	29,72
Precision	32,90	42,17	40,14	36,54
Recall	66,22	66,95	57,19	61,41

Fig. A-8: Detailed results for data sets 5000 empty scenes.



10000 samples	No Aug	Rotation	Noise	Both
IoU	31,73	0,58	1,39	-0,53
Precision	39,53	2,03	4,26	1,72
Recall	62,22	-2,70	-3,45	-5,50
IoU	30,87	33,05	34,65	32,17
Precision	38,31	44,43	49,34	45,74
Recall	62,57	56,33	53,79	52,03
IoU	29,19	32,15	35,44	29,75
Precision	35,53	41,05	46,70	37,85
Recall	62,04	59,71	59,52	58,18
IoU	35,13	31,74	29,26	31,67
Precision	44,74	39,19	35,33	40,16
Recall	62,06	62,54	63,01	59,96

Fig. A-9: Detailed results for data sets 10000 empty scenes.