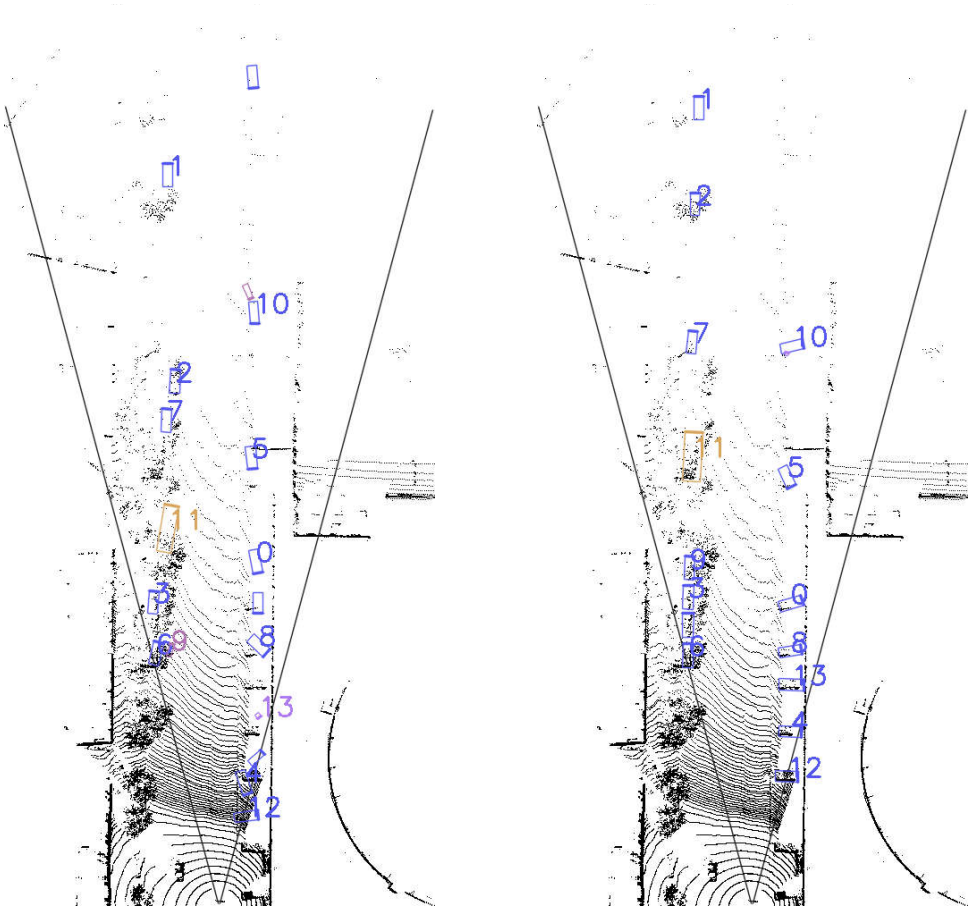
CHALMERS
UNIVERSITY OF TECHNOLOGY



Camera detections LiDAR detections

Camera-LiDAR Active Learning for Object Detecting Deep Neural Networks

Master's thesis in Computer Science - algorithms, languages and logic

ALFRED GUNNARGÅRD

DANIEL ODIN

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Camera-LiDAR Active Learning for Object Detecting Deep Neural Networks

Alfred Gunnargård

Daniel Odin

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

Camera-LiDAR Active Learning for Object Detecting Deep Neural Networks
Alfred Gunnargård, Daniel Odin

Academic Supervisor and Examiner: Lennart Svensson, Electrical Engineering
Industrial Supervisors: Dónal Scanlan, Erik Werner, Adam Tonderski, Zenuity

Master's Thesis 2020
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Bird's eye view 3D detections in LiDAR point cloud. Left: From network trained on images. Right: From network trained on LiDAR point clouds which are more accurate in position.

Typeset in LaTeX
Gothenburg, Sweden 2020

Camera-LiDAR Active Learning for Object Detecting Deep Neural Networks
Alfred Gunnargård, Daniel Odin
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Deep Neural Networks (DNNs) processing images and other types of high dimensional data need to train on *large* annotated datasets to reach usable performance. Retrieving annotations is expensive due to the required involvement of humans. In order to reduce annotation costs, it is desirable to annotate only those samples that help the DNN perform better. However, finding those samples is not trivial.

To find and annotate samples where the DNN performs poorly is one way of improving its performance. Still, finding those samples is challenging. In many cases, the output of a DNN is best evaluated when compared to an annotation made by a human. Since the annotation does not exist yet, some other type of metric is needed.

The process of choosing samples to annotate based on the performance of the DNN is called Active Learning (AL). We study AL for the task of 3D Object Detection (3DOD) in images, i.e. identifying and classifying objects in 3D space. Furthermore, we assume that all images are accompanied by a corresponding LiDAR point cloud. A LiDAR is a rotating sensor that builds a 3D model of its surroundings by measuring the time it takes for laser beams to reflect. The reflected beams result in multiple points in 3D space, referred to as a point cloud.

We quantify the performance of our DNN (referred to as primary DNN) by using a secondary DNN that detects objects in point clouds. We then compare the detections from our primary DNN to the detections of the secondary DNN and score each sample according to a discrepancy measure. The discrepancy measure quantifies how much the detections from both DNNs differ from each other for a specific sample, i.e. a pair of image and point cloud. The idea is that a sample with high discrepancy indicates that the primary DNN performs poorly due to the three-dimensional position accuracy of the secondary DNN.

We verify our approach using annotations instead of detections from a secondary DNN. According to some important metrics, we are able to perform slightly better than or on par with selecting samples randomly. However, the improved performance over random selection comes from selecting samples that contain many objects, and that raises the annotation cost. We show that a slightly modified discrepancy measure is able to perform on par with random selection while maintaining the same annotation cost.

Further research on how to properly measure this type of discrepancy is concluded to be necessary for the method to be of use in any real-case scenario. Such research includes experiments on an additional dataset.

Keywords: Machine Learning, Active Learning, 3D Object Detection, LiDAR, Deep Neural Network

Acknowledgements

We want to thank our industrial supervisors Dónal Scanlan, Erik Werner, and Adam Tonderski for their support throughout the thesis. Specifically, we want to thank Adam for his patience during technical questions and Erik for his brilliant ideas and feedback that helped us focus on the right things.

We also want to thank Lennart Svensson, our academic supervisor, for his support. We thank Lennart specifically for the support towards the end of the thesis when everything seemed dark and hopeless.

Furthermore, we thank Zenuity for letting us do our Master's Thesis at their company. During our first weeks, we were warmly welcomed and many people offered us help to get going and showed interest in our thesis. More specifically, we want to thank Brendan Barrick for his never-ending patience helping us with IT issues and Mats Nordlund for helping us to find a place to sit at AI Innovation of Sweden (until Covid-19 locked us into our homes). We also want to thank the people at AI Innovation for providing us with a nice working environment and delicious pastries (fika).

Finally, we thank Niklas Gustafsson and Gustav Rödström that did their Master's Thesis at Zenuity at the same time we did. They have kept us company during the time at Zenuity, been supportive, and provided feedback as well as suggestions to our study. Furthermore, they also provided a great opposition to our thesis with many interesting discussion subjects.

Alfred Gunnargård, Daniel Odin, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1
Introduction

The potential of self-driving cars cannot be denied. They have both promising *social* and *environmental* benefits. Self-driving cars can be safer than cars with human drivers due to shorter reaction times and constant perception around the whole vehicle. They might also be more energy-efficient, e.g. by using optimal speeds and driving behaviors, planning routes, and avoiding traffic jams by utilizing data from other cars.

The challenge of developing self-driving cars is demanding in many ways. There are many vital parts in an autonomous driving system and one of them is the perception of the surroundings. Such perception is achieved through different types of sensors mounted all around the vehicle, such as cameras and LiDAR sensors.

A LiDAR sensor (Light Detection and Ranging) builds a 3D model of its surrounding by measuring the time it takes for a laser beam to be reflected to the sensor. The generated 3D model consists of a large number of unordered points in 3D space which is why it is often referred to as a *point cloud*. Different types of sensors have different algorithms for interpreting sensor data. To extract useful information from images and point clouds, it is common to use Machine Learning (ML) algorithms.

ML algorithms are trained to recognize patterns by looking at multiple examples. If the examples are annotated, the ML algorithm learns by predicting the annotation with the purpose of being able to perform well on unseen and non-annotated data. When applying ML to images, it is common to use Deep Neural Networks (DNN) which requires a tremendous amount of annotated data in order to achieve useful performance. The cost of annotating this data can get quite large due to the required involvement of humans. It is therefore desirable to get as much information out of every annotated image as possible. One way of doing this is to use Active Learning (AL).

B. Settles manages to summarize the key idea of AL in one sentence:

> The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer labeled training instances if it is allowed to choose the training data from which it learns [3].

In most cases, only a subset of all available data has likely been annotated and some ML model has been trained on this annotated subset. It is then up to the AL algorithm to decide, based on the current state of the trained ML model, what subset of the non-annotated data is most informative and should be sent for annotation. The ML model in our case is a 3D object detecting DNN. The task of 3D Object Detection (3DOD) consists of these sub-tasks:

1. Detect all relevant objects.
2. Predict their dimensions; height, width, and length.
3. Predict their positions in 3D space.
4. Classify the objects, e.g. car, pedestrian or bicyclist.

3DOD can be done on images and other types of data such as point clouds. When doing 3DOD on a point cloud, the resulting detections have a larger accuracy in position than detections in an image. The explanation is the fact that point clouds are actual 3D representations whereas images lack explicit data on distance to objects. However, a point cloud is often sparse at distances far away from the vehicle. Thus, 3DOD on images will find objects further away than what 3DOD on point clouds can handle.

Now, suppose a 3DOD DNN makes perfect predictions on a non-annotated image. Annotating the image will be unnecessary since the DNN will not learn anything new from the annotation. Therefore, images, where the DNN is inaccurate, are more rewarding to annotate. However, finding images where the DNN is inaccurate is a non-trivial task.

Comparing with predictions from DNNs that use data from other sensors, where the predictions are more accurate, could help to find the images where the DNN is inaccurate. Intuitively, since images are 2D representations of the world, it is harder to predict an object's position in 3D space by looking at an image than looking at a point cloud. Therefore, a DNN trained on point clouds can help find images where the DNN trained on images is inaccurate.

1.1 Thesis Objective

The objective of this thesis is to investigate the possibility of utilizing the difference between LiDAR sensors and cameras to do AL for 3DOD in images. In particular, by having each sample in the dataset consisting of an image and a point cloud captured at the same time, we can compare the detections obtained from two different DNNs.

We investigate how to use a primary DNN for 3DOD in images in parallel with a secondary DNN for 3DOD in point clouds. We call the primary DNN $\mathcal{M}_P$ (where P stands for primary) and the secondary DNN $\mathcal{M}_S$ (where S stands for secondary). The detections from $\mathcal{M}_P$ are compared to the detections from $\mathcal{M}_S$ in order to calculate a camera-LiDAR detection discrepancy measure. This measure is then used to score images. In AL, the score is used to select images for annotation that have a large discrepancy, in order to improve the performance of $\mathcal{M}_P$. The intuition behind why this is worth investigating is that the samples with the largest discrepancy value are assumed to be those where $\mathcal{M}_P$ is wrong or uncertain.

1.2 Limitations

The main goal of the thesis is to develop an AL algorithm for 3DOD and not a DNN architecture. Therefore, we limit ourselves to use existing implementations of architectures. The implementations that we use are provided by Zenuity and we only go into brief details regarding the architectures.

1.3 Contributions

We contribute with a definition of a discrepancy measure that quantifies how two sets of detected objects differ from each other. Furthermore, we present results showing that the proposed discrepancy measure can be used for scoring samples to do AL. The method performs better than or on par with selecting samples at random but raises the annotation cost. The presented results further contribute with the analysis that the proposed discrepancy measure can be modified to maintain the annotation cost. However, the modified measure fails to do so while performing better than random selection. Instead, the modified measure only performs on par with random selection.

Discussed are ways to improve the method with the goal of outperforming random selection while maintaining the annotation cost, which is crucial for the method to be useful in a real-case scenario.

1.4 Related Work

When doing AL for ML in general, the most straight forward approach is to use some kind of *uncertainty sampling* [3]. Uncertainty sampling is a method where the samples selected for annotation are those where the model is least confident in its predictions. The samples selected are often the samples where the model is not confident in classification. The approach is feasible for 3DOD, but the classification is just one part of a prediction in 3DOD, as explained above. Therefore, it would be beneficial to include the other parts of a prediction in the selection process. The uncertainty for the other parts of a 3DOD prediction is difficult to find since 3DOD DNNs do not exhibit uncertainties for those parts. This thesis proposes a method with the potential to include samples where the model is wrong or uncertain for the other components as well.

Another well-established approach is Query by Committee where samples are selected based on the disagreement between a committee of models where all models are trained on the same dataset [4]. One could see our approach as a committee of two models. However, even though both models train on samples captured at the same moment, the samples are not of the same format. According to the definition of Query by Committee, the models should train on the same dataset in the same format. However, in our case, $\mathcal{M}_P$ is using images, and $\mathcal{M}_S$ is using point clouds.

Our approach can also be compared to [5] where a *View Divergence Score* is proposed as a score of informativeness in AL for semantic segmentation. The View Divergence Score determines what samples are the most informative based on how

the predictions differ between different views of the same scene. The different views, in this case, are images taken from different angles. This is similar to what we do but since the scope of [5] is semantic segmentation, it does not cover how to measure the discrepancy between detections. Neither do they use point clouds but rather only images.

Another approach uses AL for training an OD DNN for images and proposes two metrics for the informativeness of a sample [6]. The two metrics are designed to model the uncertainty of the detections. However, these metrics do not compare multiple predictions coming from different sources (or views) to do AL, as our discrepancy measure does. Furthermore, [6] focus solely on 2DOD while our method is developed for 3DOD.

Using both LiDAR and cameras in combination with AL has been done by [7]. However, the actual AL algorithm in [7] does not benefit from the differences between the two types of sensors, as ours does. Furthermore, only the DNN for OD in the point clouds is trained using the proposed AL algorithm. We do the opposite - train the DNN for OD in images.

1.5 Thesis Outline

Chapter 2 explains the theory and all the necessary components to understand the thesis. The chapter includes a short description of ML and DNN, the fundamental concept of AL as well as a description of the LiDAR and OD, the Hungarian method for solving the 2D Assignment Problem, and evaluation metrics for OD. In Chapter 3, our method is described including a description of the dataset and the models. The discrepancy measure between detections from two DNNs $\mathcal{M}_P$ and $\mathcal{M}_S$ is described in detail. Chapter 4 contains the settings used in the experiments whose results are presented in Chapter 5. The results are discussed in Chapter 6 with a conclusion in Chapter 7.

2
Theory

This chapter covers the technical details necessary to understand the conducted research. We begin by shortly describing Machine Learning and Artificial Neural Networks in Section 2.1 to motivate the reason for Active Learning. The concept of Active Learning and how traditional Active Learning algorithms work is described in Section 2.2. Then, the LiDAR sensor is described in Section 2.3 followed by 2D and 3D Object Detection in Section 2.4. We also describe the key parts of our method which are the Hungarian Method for solving the 2D Assignment Problem in Section 2.5 and Intersection over Union in Section 2.6. Lastly, evaluation metrics for Object Detection are explained in Section 2.7.

2.1 Machine Learning

Machine Learning (ML) is a common technique for solving complex tasks such as detecting objects in images. The task is solved by letting the ML algorithm learn by training on multiple samples. If the samples are *annotated*, the task is called *supervised learning*. In supervised learning, the annotation is the desired output of the ML algorithm.

The goal in supervised learning is to learn the output y for a specific input x such that the ML algorithm can accurately predict the output for unseen and non-annotated data. More specifically, the ML algorithm tries to approximate a function $y = f^*(x)$ with $f(x; \theta)$ by adjusting some parameters θ. A common way of approximating such a function is by Artificial Neural Networks.

2.1.1 Artificial Neural Networks

Inspired by biological neural networks, e.g a human brain, Artificial Neural Networks can solve advanced computational tasks by using a large number of connected processors, referred to as artificial *neurons*. The neurons serve as single units where each unit act as a function taking a vector as input and outputs a scalar [8]. All the connected neurons together build up to a complex function $\hat{\mathbf{y}} = f(\mathbf{x}; \boldsymbol{\theta})$ that approximates $\mathbf{y} = f^*(\mathbf{x})$. The parameters $\boldsymbol{\theta}$ often consist of weights $\mathbf{w}$ and biases $\mathbf{b}$.

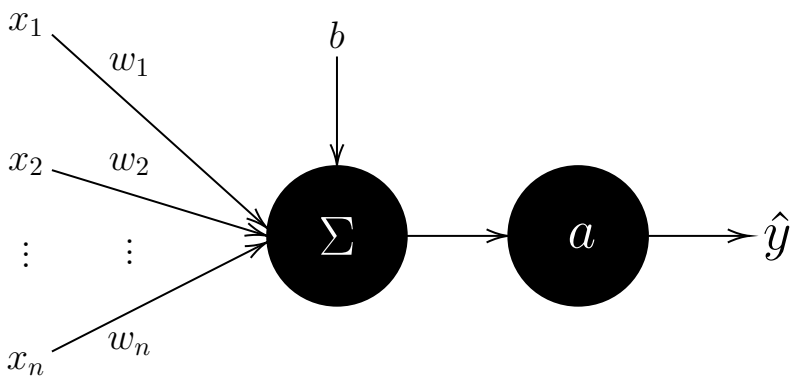


Figure 2.1: A visualisation of a neuron. Each input x_i is multiplied with a weight w_i and summed together. A bias term b is added to the sum. The final sum is the input for a non-linear activation function a to produce the final output $\hat{y}$.

A visualization of a single neuron can be seen in Figure 2.1. A neuron takes a vector $\mathbf{x} = (x_1, ..., x_n)^T$ as input and outputs a single value $\hat{y}$. The mathematical definition of a neuron is

$$\hat{y} = a(\sum_{i=1}^{n} w_i x_i + b) \tag{2.1}$$

where $\mathbf{w} = (w_1, ..., w_n)^T$ and b are learnable parameters referred to as weights and bias and a is a non-linear activation function. Commonly used activation functions are

$$\text{Rectified Linear Unit :} \qquad \text{ReLU}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{otherwise} \end{cases} \tag{2.2}$$

$$\text{Sigmoid :} \qquad \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

$$\text{Hyperbolic tangent :} \qquad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.4}$$

Using a large number of neurons, the neural network can approximate many complex continuous functions. The activation functions allow the network to learn non-linearities. Without the activation functions, the network is just one linear operation regardless of how many neurons the network consists of.

The parameters $\boldsymbol{\theta}$ are trained by minimizing the total loss over the whole training dataset using a loss function L, i.e.

$$\min_{\boldsymbol{\theta}} \sum_i L(\mathbf{y}_i, f(\mathbf{x}_i; \boldsymbol{\theta})) \tag{2.5}$$

where $\mathbf{y}_i$ is the desired output and $\hat{\mathbf{y}}_i = f(\mathbf{x}_i; \boldsymbol{\theta})$ is the output from the neural network for sample i. The loss essentially measures how far the prediction $\hat{\mathbf{y}}_i$ is from the truth $\mathbf{y}_i$. The gradient of L with respect to $\boldsymbol{\theta}$ is used to optimize $\boldsymbol{\theta}$ by

taking small steps in the opposite direction of the gradient. We refer to [8] for details.

The most fundamental network is the *fully connected feedforward network* [8]. In a fully connected feedforward network, the neurons are connected in several layers, where all neurons are connected between consecutive layers. An example of a fully connected feedforward network can be seen in Figure 2.2. The word feedforward means the input $\mathbf{x}$ is evaluated through all the layers in f, as an ordinary function, without any intermediate feedback to itself.

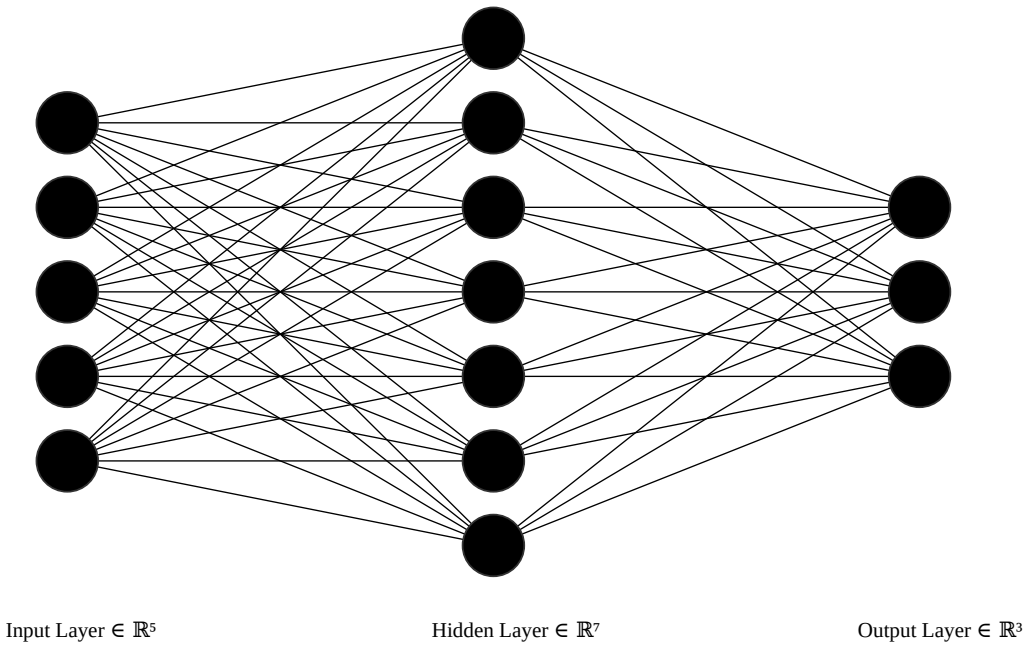


Figure 2.2: An example of a fully connected feedforward network consisting of input layer $\mathbf{x} \in \mathbb{R}^5$ and output layer $\hat{\mathbf{y}} \in \mathbb{R}^3$. The hidden layer consists of 7 neurons. Each connection corresponds to a learnable weight.

The fully connected feedforward network consists of an input layer that constitutes input $\mathbf{x}$, up to several hidden layers, and an output layer that constitutes output $\hat{\mathbf{y}}$. Each layer outputs the input for the next layer. The word hidden layer comes from that the training data does not show the desired output from these kinds of layers. The number of layers is often referred to as the depth of the neural network.

For complex tasks, such as detecting objects in images, it is common to use neural networks with large depth, i.e. a tremendous amount of layers. Due to the large depth, these kinds of networks are called Deep Neural Networks.

2.1.2 Deep Neural Networks

A Deep Neural Network (DNN) can consist of hundreds of layers which is crucial for approximating complicated functions. For example, detecting objects in images is demanding in many ways. The DNN needs to learn what the different objects look like as well as what environments the objects interact with. Therefore, to perform the object detection task well, a tremendous amount of features need to be learned.

An image can consist of millions of pixels. Due to the millions of pixels, if an image is used as input to a fully connected feedforward network, the input layer (see Figure 2.2) will consist of millions of neurons. Therefore, the number of parameters will be huge since each connection between neurons has an associated weight. Instead, one can use a *convolutional neural network* [8]. In a convolutional layer, only nearby neurons, i.e. nearby pixels for images, are connected in the neural network. The same weights are used in all neighborhoods, called weight sharing. However, even for convolutional networks, the number of parameters can still be large due to many layers.

Many popular DNNs for solving image tasks, as well as the DNNs used in this thesis, are built upon an efficient DNN called ResNet. ResNet is a convolutional neural network using a successful technique called residual learning. We refer to [9] for details. ResNet-152 consists of 152 layers and millions of learnable parameters. To optimize all the millions of parameters, a great number of annotated training data is needed.

The annotation process is tedious, expensive, and requires a human. If a DNN makes a perfect prediction on a non-annotated sample, it is most likely not informative to the DNN to train on. Since a large number of annotated training data is needed, it is important to annotate samples that are informative for the DNN. To efficiently choose data for annotation which is believed to be informative, Active Learning can be used.

2.2 Active Learning

The core of Active Learning (AL) is to choose what data to use for training an ML model. Based on some *informativeness score*, the AL algorithm (also known as the *learner*) can select data which the model believes is informative. The informativeness score attempts to measure how beneficial a non-annotated sample would be to annotate. If the non-annotated sample is considered informative, the learner queries for an annotation and adds the newly annotated sample to the training set.

There are mainly three different types of AL: *membership query synthesis*, *stream-based selective sampling* and *pool-based active learning* [3]. Membership query synthesis lets the learner create its own data which it believes the ML model will benefit the most from. Thereafter the learner queries for annotations of the newly created data. Stream-based selective sampling sequentially analyses a non-annotated dataset. For each sample, the learner makes a decision based on the informativeness score whether to keep it (i.e. query for its annotation) or discard it. Lastly, pool-based active learning is much like stream-based selective sampling with the difference that it analyses the whole non-annotated dataset before deciding what

samples to annotate [3].

Membership query synthesis does not suit complex image tasks, such as 3DOD, since synthesizing realistic images can be difficult. In stream-based selective sampling, only one sample is drawn at a time. In many real-world cases, large collections of non-annotated data can be collected at once which makes it possible to analyze all the samples.

In this thesis, we assume a large collection of non-annotated data can be collected at once and added to a pool of non-annotated data. We also assume that a batch of non-annotated samples are selected and sent for annotation at the same time and added to an existing pool of annotated data. Therefore, we base our method on pool-based active learning. Figure 2.3 shows an overview of how pool-based AL works. The target model in the figure is a DNN. The generic framework for pool-based active learning consists of two datasets $\mathcal{A}$ and $\mathcal{N}$ where the first has annotations and the second does not. We call them the *annotated* and *non-annotated* datasets respectively. We have a budget of b samples and denote the target ML model (the model that is trained by the learner) by $\mathcal{M}$. The AL algorithm is explained step by step below and each step has a reference to the arrows in Figure 2.3.

1. Train $\mathcal{M}$ on $\mathcal{A}$ (purple arrow).
2. Apply $\mathcal{M}$ on $\mathcal{N}$ and store the output (yellow arrow).
3. Score the output from step 2 using some informativeness score (yellow arrow).

4. Pick the b samples from $\mathcal{N}$ with highest score (yellow arrow) and send for annotation (blue arrow).
5. Add the annotated data to $\mathcal{A}$ and remove it from $\mathcal{N}$ (turquoise arrow).
6. Repeat from step 1.

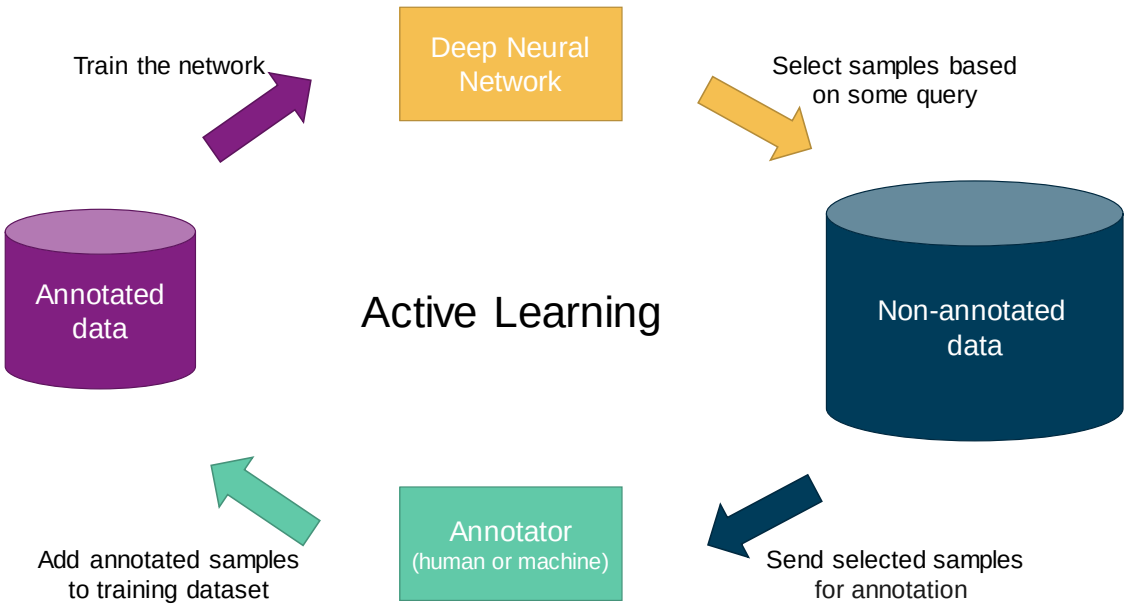


Figure 2.3: Overview of a pool-based AL algorithm for DNNs.

The main difficulty for AL is to design an informativeness score that correlates well with what the model needs to learn. This is highly dependent on both the task that should be solved and the available underlying data. Since designing an informativeness score is not trivial, we look at a few examples of how the informativeness of a sample can be measured in the next subsection.

2.2.1 Examples of Informativeness Scores

One of the most straight forward query approaches is *Uncertainty Sampling* [3]. In Uncertainty Sampling, the samples that the model is most uncertain about are chosen for annotation. The most uncertain samples are given a large score and assumed to be the most informative. The uncertain samples are samples where the model is not confident in its predictions.

For classification tasks, the output is a list of class probabilities. To measure uncertainty in classification tasks, *entropy* is suitable [3]. The samples with large entropy are selected for annotation. For a random variable Y with a set $\{y_i\}$ of possible outcomes, entropy is defined as

$$\text{Entropy}(Y) = -\sum_i P(Y = y_i) \log P(Y = y_i) \tag{2.6}$$

where $P(Y = y_i)$ is the probability that y_i occur. For a random variable, the entropy is largest if the probability for all outcomes is equal which means that we cannot confidently tell what the outcome will be. Therefore, the outcome is uncertain.

Consider the task of classifying images of cats and dogs as an example. Suppose the output of the model is 52% probability that an image contains a cat and 48% probability that the image contains a dog. Even though the probability of being a cat is larger than being a dog, the similarity indicates that the classification model is uncertain. There could be unseen features in the image that makes it uncertain to predict the class. Therefore, the model could benefit from training on the sample in order to learn new cat and dog features. By contrast, if the probability of a dog is 99% for a decently trained ML algorithm, the image likely contains a dog, in which case annotating this image does not significantly improve the performance of the algorithm.

Another query approach is *Query by Committee* [3]. A committee of models trains on the same annotated data and the samples selected are based on the disagreement between the models' outputs. If the disagreement between the models' output is large, the informativeness score is large. For classification tasks, the disagreement can be measured using *vote entropy* [3]. For a set $\{y_i\}$ of possible classes and C number of committee models, vote entropy for sample x is defined as

$$\text{VoteEntropy}(x) = -\sum_i \frac{V(y_i; x)}{C} \log \frac{V(y_i; x)}{C} \tag{2.7}$$

where $V(y_i; x)$ is the number of votes, i.e. number of models predicting class y_i for sample x. The disagreement is large if the votes are evenly distributed over

the different classes. Evenly distributed votes lead to all $\frac{V(y_i;x)}{C}$ terms being similar which corresponds to large entropy, as explained above.

Compared to Uncertainty Sampling and Query by Committee, a more different query approach is *Expected Model Change*. Expected Model Change measures how much impact the sample has on the current model if it is included in the training set. The sample is assumed to be informative if it makes a great impact on the model, i.e. make large changes to its parameters.

An example of Expected Model Change used for classification tasks is *Expected Gradient Length*. Expected Gradient Length calculates the length of the loss function gradient. Since the sample is non-annotated, we calculate the average length of the gradient over all possible classes. Let $\{y_i\}$ be a set of all possible classes and $\nabla L(y_i, f(x;\theta))$ be the gradient for the loss between y_i and the prediction for $f(x;\theta)$ using the current model parameters θ. Expected Gradient Length for sample x is defined as

$$\text{ExpectedGradientLength}(x) = \sum_i P(y_i|x;\theta)||\nabla L(y_i, f(x;\theta))||_2 \tag{2.8}$$

where $P(y_i|x;\theta)$ is the probability of x being class y_i and $||\cdot||_2$ is the L2-norm. Empirical studies show that Expected Gradient Length works well for some tasks, but can be computationally expensive if both the number of possible classes and the feature space is large [3].

2.2.2 Evaluating an Active Learning Algorithm

An AL algorithm should be evaluated before used in any real-case scenario due to the annotation procedure being both time-consuming and expensive. The selected samples sent to the annotator must be of some value for the model, otherwise, the time spent on annotation is wasted. When evaluating an AL algorithm, the human annotator is commonly simulated. Instead of having a human annotator, the non-annotated dataset $\mathcal{N}$ is not used and the annotated dataset is divided into two parts. The first part is unchanged and interpreted as the annotated dataset $\mathcal{A}$. The second part acts as $\mathcal{N}$ by hiding the annotations. The human annotator is simulated by revealing the annotations for some data from the second part and moving it to the first part.

2.3 LiDAR Sensor

Our AL method measures the informativeness of a sample by using predictions based on data from a LiDAR sensor. We refer to Chapter 3 for details on our method.

A LiDAR sensor (Light Detection and Ranging) can be mounted on a self-driving car to achieve perception around the vehicle while moving. The LiDAR measures the distance to objects by measuring the time it takes for a laser beam to be reflected to the sensor. The laser beams build a 3D model of the surroundings, consisting of a large number of unordered points. Due to the large number of unordered points, the 3D model is referred to as *point cloud*.

An example of a point cloud can be seen in Figure 2.4. The point cloud is visualized in *bird's eye view*, meaning every point is projected onto the ground and shown from above. How high the points are above the ground is not shown in the figure. Objects in the point cloud are clearly marked with blue and orange boxes, where blue indicates a car and orange a truck. The objects are detected by an Object Detection DNN using point clouds as input. Predicting the position in 3D space using a point cloud is less difficult than predicting the position using an image. An image lack explicit data on distance to objects, whereas a point cloud is an actual 3D representation.

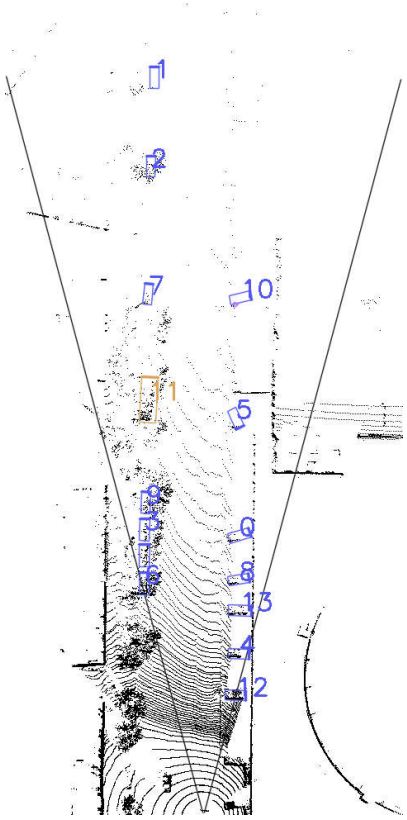


Figure 2.4: Example of LiDAR point cloud in bird's eye view with detections of objects.

2.4 Object Detection

Object Detection (OD) is a technique used for detecting objects of certain classes, such as cars and/or pedestrians. It is commonly applied to images, but can also be used in other models of the world, such as point clouds. Objects can be detected both in 2D (images) and 3D (images and point clouds).

An example of 2DOD can be seen in Figure 2.5. A detected object is bounded in a box (a *bounding box*) and the object in the box is predicted with a class [10]. The 2D bounding box expresses where the object is located *in the image*.

In 3DOD [11], by contrast to 2DOD, the bounding box is three-dimensional and expresses where the object is located in world coordinates, relative to the *ego vehicle*. The ego vehicle is the vehicle where all the sensors are mounted and is the vehicle from where the image is captured. As in 2DOD, the object is predicted with a class. The 3D bounding box is defined in 3D space and is commonly expressed by position (center point coordinates of bounding box bottom plane), dimension (height, width, length), and rotation around the three axes. An example of 3DOD can be seen in Figure 2.6. Since the boxes are defined in 3D space, this definition can apply to 3DOD in both point clouds and images.

Figure 2.5: Example of 2D object detection.

Figure 2.6: Example of 3D object detection. Predicted location together with dimension and facing direction in 3D space is visualized in bird's eye view to the right.

2.5 2D Assignment Problem

The idea of this thesis is to find images where a DNN trained on images gives different predictions than a DNN trained on point clouds. We compare detections from a primary DNN $\mathcal{M}_P$ (where P stands for primary) trained on images with detections from a secondary DNN $\mathcal{M}_S$ (where S stands for secondary) trained on point clouds. The comparison is done by calculating a discrepancy between detections.

To calculate a discrepancy measure between detections outputted from $\mathcal{M}_P$ and $\mathcal{M}_S$, one approach is to calculate the discrepancy per object. If the discrepancy is calculated per object, each detection from $\mathcal{M}_P$ needs to be matched to a detection in $\mathcal{M}_S$ and vice versa. Both detections in a match are associated to the same real-world object. The matching can be solved by solving the 2D Assignment Problem (2DA) [12].

Let $G = \langle V, U; E \rangle$ be a weighted bipartite graph where V and U consist of vertices, E consist of edges and each edge $e \in E$ is assigned a weight. Furthermore, let $|V| \leq |U|$. An example of a bipartite graph can be seen in Figure 2.7. The 2DA consists of finding the minimum sum of weights such that each vertex $v \in V$ is connected to exactly one vertex $u \in U$. For example, if V correspond to jobs, U correspond to machines and each edge $e \in E$ has a running time, we want to find the matchings of jobs and machines where the total running time is minimized, all jobs are run and one machine is assigned exactly one job. An approach to solving the 2DA is the Hungarian method [13].

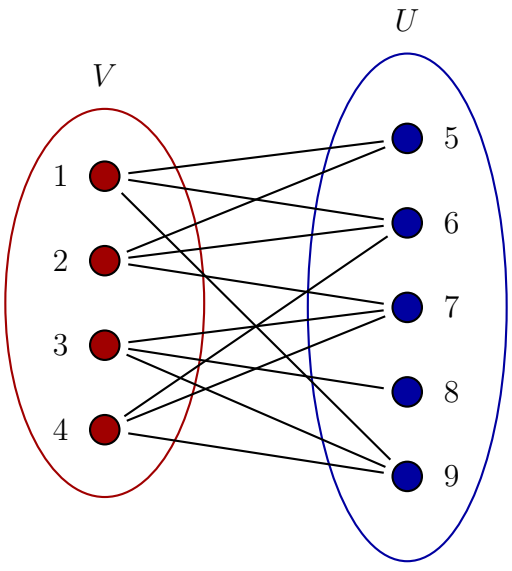


Figure 2.7: An example of a bipartite graph. The 2D assignment problem consists of finding edges such that each vertex in V (1-4) is connected to exactly one vertex in U (5-9) and the sum of weights is minimized.

2.5.1 Hungarian Method

The Hungarian method can solve the 2DA in $\mathcal{O}(n^3)$ time [14], where $n = |U|$. The 2DA can be expressed with a matrix M where the elements are the weights of E. The rows correspond to V and the columns correspond to U. Therefore, index (i, j) of M contains weight m_{ij} for connecting the ith vertex in V to the jth vertex in U. If $|V| < |U|$, $|U| - |V|$ dummy rows with zeros need to be added such that $|V| = |U|$. After the dummy rows are added, the 2DA can be expressed as

$$\text{minimize} \quad \sum_{(i,j)\in E} y_{ij} m_{ij} \tag{2.9a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} y_{ij} = 1, \quad \forall j \in \{1, ..., n\}, \tag{2.9b}$$

$$\sum_{j=1}^{n} y_{ij} = 1, \quad \forall i \in \{1, ..., n\}, \tag{2.9c}$$

$$y_{ij} \in \{0, 1\}, \quad \forall i, j \in \{1, ..., n\} \tag{2.9d}$$

where $y_{ij} = 1$ if the ith vertex in V is assigned the jth vertex in U. The constraints in (2.9b) and (2.9c) ensures that each vertex $v \in V$ is assigned exactly one vertex $u \in U$. The constraint (2.9d) ensures that an assignment cannot be split. In the example with machines and jobs, the constraint ensures that one job cannot be split and run by two machines.

We will now describe how the Hungarian method is executed. After that, we will take a look at the mathematical details as to why the solution is optimal. We use

$$M = \begin{pmatrix} 4 & 1 & 5 \\ 7 & 5 & 6 \\ 8 & 5 & 8 \end{pmatrix} \tag{2.10}$$

as an example problem instance. Let $V = \{v_1, v_2, v_3\}$ and let $U = \{u_1, u_2, u_3\}$.

1. (a) For each row r in M:
   Subtract the minimum value of r from every element in r.
   Minimum values are marked with a circle.

$$\begin{pmatrix} 4 & \textcircled{1} & 5 \\ 7 & \textcircled{5} & 6 \\ 8 & \textcircled{5} & 8 \end{pmatrix} \begin{matrix} \leftarrow -1 \\ \leftarrow -5 \\ \leftarrow -5 \end{matrix} \quad \Rightarrow \quad \begin{pmatrix} 3 & 0 & 4 \\ 2 & 0 & 1 \\ 3 & 0 & 3 \end{pmatrix} \tag{2.11}$$

   (b) For each column c in M:
   Subtract the minimum value of c from every element in c.
   Minimum values are marked with a circle.

$$\begin{array}{c} \begin{matrix} -2 & -0 & -1 \\ \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{pmatrix} 3 & \textcircled{0} & 4 \\ \textcircled{2} & 0 & \textcircled{1} \\ 3 & 0 & 3 \end{pmatrix} \end{array} \quad \Rightarrow \quad \begin{pmatrix} 1 & 0 & 3 \\ 0 & 0 & 0 \\ 1 & 0 & 2 \end{pmatrix} \tag{2.12}$$

2. (a) Cover all zeros in M using minimum number of vertical and horizontal lines.

$$\begin{pmatrix} 1 & 0 & 3 \\ \cancel{0} & \cancel{0} & \cancel{0} \\ 1 & 0 & 2 \end{pmatrix} \tag{2.13}$$

   (b) If the number of lines is equal to number of rows in M, go to step 3, otherwise proceed.

   (c) Add the minimum value of all the uncovered (*marked with circles*) elements to all elements covered by both a vertical and a horizontal line (*marked with square*).

$$\begin{array}{c} \begin{matrix} & +1 & \\ & \downarrow & \end{matrix} \\ \begin{pmatrix} \textcircled{1} & 0 & \textcircled{3} \\ \cancel{0} & \boxed{0} & \cancel{0} \\ \textcircled{1} & 0 & \textcircled{2} \end{pmatrix} \end{array} \quad \Rightarrow \quad \begin{pmatrix} 1 & 0 & 3 \\ 0 & \boxed{1} & 0 \\ 1 & 0 & 2 \end{pmatrix} \tag{2.14}$$

   (d) Subtract the minimum value of all the uncovered elements from all the uncovered elements (*marked with circles*).

$$\begin{array}{c} \begin{matrix} -1 & -1 \\ \downarrow & \downarrow \end{matrix} \\ \begin{pmatrix} \textcircled{1} & 0 & \textcircled{3} \\ \cancel{0} & 1 & \cancel{0} \\ \textcircled{1} & 0 & \textcircled{2} \end{pmatrix} \end{array} \quad \Rightarrow \quad \begin{pmatrix} \textcircled{0} & 0 & \textcircled{2} \\ 0 & 1 & 0 \\ \textcircled{0} & 0 & \textcircled{1} \end{pmatrix} \tag{2.15}$$

   (e) Remove the lines and go to step 2.

3. Find the solution by systematically picking zeros covered by either a vertical or a horizontal line but not both. Also, only pick one element from each row and column. If the solution is valid, the solution is also optimal.
The lines in 2.16 comes from doing step 2a again. A valid solution is marked with squares at indices $(1,1)$*,* $(2,3)$ *and* $(3,2)$*. Therefore, the resulting matchings are* (v_1,u_1)*,* (v_2,u_3) *and* (v_3,u_2)*.*

$$\begin{pmatrix} \cancel{0} & \cancel{0} & 2 \\ \cancel{0} & \cancel{1} & \cancel{0} \\ \cancel{0} & \cancel{0} & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \boxed{0} & 0 & 2 \\ 0 & 1 & \boxed{0} \\ 0 & \boxed{0} & 1 \end{pmatrix} \tag{2.16}$$

$$M = \begin{pmatrix} \boxed{4} & 1 & 5 \\ 7 & 5 & \boxed{6} \\ 8 & \boxed{5} & 8 \end{pmatrix} \quad \Rightarrow \quad \text{total sum of weights: } 4+6+5=15 \tag{2.17}$$

The Hungarian method modifies M in order to find an optimal solution. The mathematical reasoning to why the Hungarian method gives an optimal solution is as follows. Let $\{a_i\}_{i=1}^n$ and $\{b_j\}_{j=1}^n$ be values that modifies M. The values modify M without changing (2.9a) as

$$\sum_{(i,j)\in E} y_{ij}m_{ij} = \sum_{(i,j)\in E} (m_{ij} - a_i - b_j + a_i + b_j)y_{ij} \tag{2.18}$$

$$= \sum_{(i,j)\in E} (m_{ij} - a_i - b_j)y_{ij} + \sum_{i=1}^{n} a_i \sum_{j=1}^{n} y_{ij} + \sum_{j=1}^{n} b_j \sum_{i=1}^{n} y_{ij} \tag{2.19}$$

$$= \sum_{(i,j)\in E} (m_{ij} - a_i - b_j)y_{ij} + \sum_{i=1}^{n} a_i \cdot 1 + \sum_{j=1}^{n} b_j \cdot 1 \tag{2.20}$$

$$= \sum_{(i,j)\in E} (m_{ij} - a_i - b_j)y_{ij} + \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j \tag{2.21}$$

where (2.20) is obtained by the constraints (2.9b) and (2.9c). The Hungarian method finds an optimal solution by modifying M with $\{a_i\}_{i=1}^n$ and $\{b_j\}_{j=1}^n$, without violating the constraints, such that $m_{ij} - a_i - b_j = 0$ for (i,j) where $y_{ij} = 1$. The first term, $\sum_{(i,j)\in E}(m_{ij} - a_i - b_j)y_{ij}$, evaluates to 0, since $y_{ij} = 0$ for all other (i,j), meaning the resulting objective value is $\sum_{i=1}^n a_i + \sum_{j=1}^n b_j$. The solution is optimal since the resulting objective value is independent of y_{ij} and the term including y_{ij} is minimal since the term is evaluated to 0. We refer to [15] and [16] for details.

2.6 Matching Score

In order to match detections between $\mathcal{M}_P$ and $\mathcal{M}_S$, a score for a potential match is needed in order to create M used by the Hungarian method to solve the 2DA. We define a good match as a large Intersection over Union (IoU) between 2D bounding box detections. The IoU between two sets A and B is defined as

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.22}$$

and measures the similarity between two sets. In 2DOD, IoU between bounding boxes measures area of overlap (in pixels) divided by area of union (in pixels). A visualization can be seen in Figure 2.8. In 3DOD, IoU is calculated with volume (in cubic meters) instead of area.

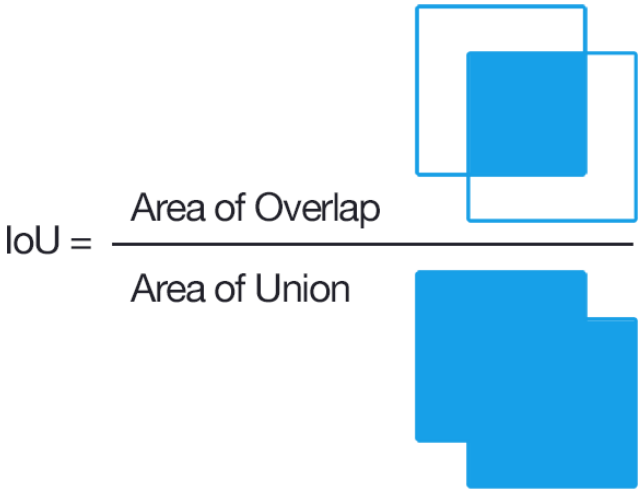


Figure 2.8: A visual equation for Intersection over Union (Jaccard Index) [1].

2.7 Evaluation Metrics

Evaluation metrics are used to evaluate and compare different AL algorithms. To evaluate an AL algorithm for OD, two commonly used metrics for evaluating both 2DOD and 3DOD are F1 score and Average Precision (AP). Both metrics are calculated using

$$\begin{aligned} p &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ r &= \frac{\text{TP}}{\text{TP} + \text{FN}} \end{aligned} \qquad \begin{aligned} \text{TP} &= \text{Number of true positives} \\ \text{FP} &= \text{Number of false positives} \\ \text{FN} &= \text{Number of false negatives} \end{aligned} \tag{2.23}$$

where p and r denote *precision* and *recall*, respectively. In OD, precision measures how accurate the detections are by the percentage of correct detections (true positives). IoU in 2DOD and 3DOD is often used to determine whether a detection is a true positive or false positive by comparing to the annotated object. If IoU is above a certain threshold, e.g. 0.5, the detection counts as a true positive [17].

Recall is a measure of completeness. In OD, recall measures what percentage of all the annotated objects are detected. An illustration of precision and recall can be seen in Figure 2.9.

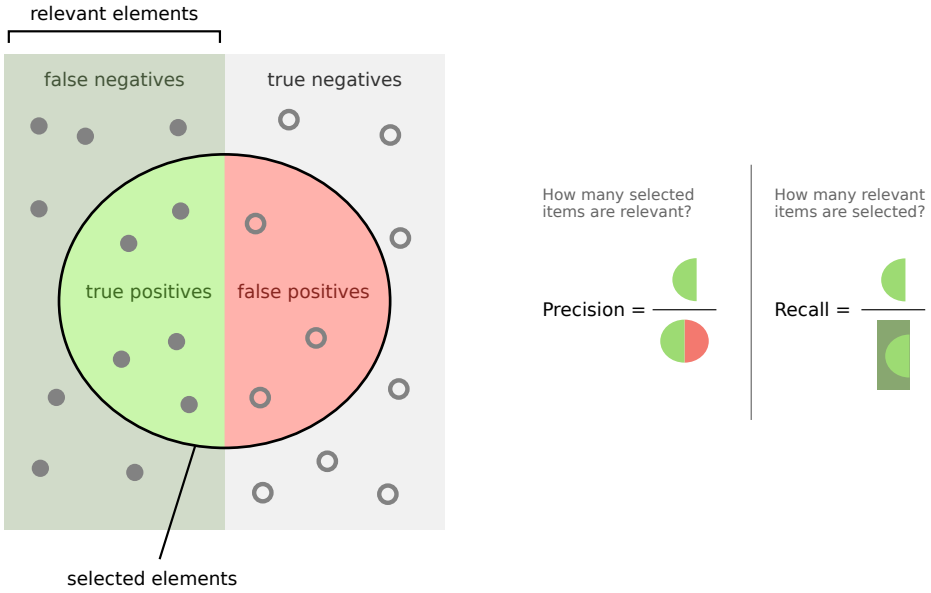


Figure 2.9: Illustration of precision and recall which are common components for evaluation metrics in object detection [2].

2.7.1 F1 Score

F1 score is defined as the harmonic mean between precision and recall, i.e.

$$\mathrm{F1} = \frac{2}{(1/p) + (1/r)} = 2 \cdot \frac{p \cdot r}{p + r} \tag{2.24}$$

and ranges between 0 and 1 [18]. With perfect precision and recall, F1-score reaches its best value, 1.

2.7.2 Average Precision

To calculate AP, precision values over the recall range 0 to 1 need to be calculated. In other words, a precision function $p(r)$ taking recall value as input should output precision value. The function $p(r)$ is called the precision-recall curve and is estimated from the predictions for all the samples in the data set.

All the predictions are sorted by their *confidence level* in descending order. In OD, the confidence level corresponds to the class prediction of the detection and is the probability of being a particular class. When all the predictions are sorted, each prediction is determined whether it is a true positive or not, which means using an IoU threshold between the detection and the annotated object in OD. Each prediction is assigned a rank depending on their confidence level. Precision and recall are calculated for each rank, where only the current rank and lower rank predictions are taken into account.

In the example in Table 2.1, there are 10 positives to be predicted in the whole dataset. All predictions are sorted by their confidence level in descending order. Rank 1 refers to the prediction with largest confidence. Each prediction is determined whether it is a true or false positive. Up to rank 5, only 3 of the predictions are true positives, which means the precision value is $3/5 = 0.6$ for that rank. The recall value depends on the total number of positives in the dataset and would in this example be $3/10 = 0.3$ for rank 5.

Confidence level	Rank	Correct prediction	Precision $p(r)$	Recall r
0.98	1	True	1	0.1
0.95	2	False	0.5	0.1
0.93	3	False	0.33	0.1
0.9	4	True	0.5	0.2
0.86	5	True	0.6	0.3

Table 2.1: An example of calculating recall and precision over a small dataset containing a total of 10 positives to be predicted.

AP measures average precision over the whole recall range 0 to 1 [17], defined by the area under the precision-recall curve, i.e.

$$\text{AP} = \int_0^1 p(r)dr. \tag{2.25}$$

In reality, the precision-recall curve is not continuous. Instead, the precision-recall curve is estimated as above and AP is commonly calculated by interpolation. A commonly used OD benchmark is the VOC2007 challenge. We refer to [19] for details. In the challenge, AP is calculated as the mean interpolated precision at $r = 0, 0.1, ..., 1$, i.e.

$$\text{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1, ..., 1\}} p_{interpolated}(r) \tag{2.26}$$

where the interpolated precision is

$$p_{interpolated}(r) = \max_{r' \geq r} p(r'). \tag{2.27}$$

The IoU threshold for determining true positives in VOC2007 is set to 0.5. Also, if there are several detections for the same object, only the detection with the largest confidence is counted as a true positive and the rest as false positives.

When there are several classes in the dataset, it is common to calculate AP for every class individually. The class AP's are averaged to obtain Mean Average Precision [17]. Even though it could be confusing, mAP is often referred to as AP.

2.7.3 Mean Absolute Error

To evaluate the accuracy of the 3D bounding box, Mean Absolute Error (MAE) can be used. The general definition for MAE is

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{2.28}$$

where $\hat{y}_1, ..., \hat{y}_N$ are predictions and $y_1, ..., y_N$ are targets. For 3DOD, the predictions are all the true positives and compared to their corresponding annotations. MAE can be used to see the accuracy of each 3D bounding box attribute separately, for example the accuracy of the bounding box height.

3
Method

This chapter introduces how we select the samples for annotation in AL. The samples are selected based on the discrepancy between detections outputted from two different DNN's, the primary DNN $\mathcal{M}_P$ and the secondary DNN $\mathcal{M}_S$. The objective is to increase the performance of $\mathcal{M}_P$ by selecting samples that are assumed to be those where $\mathcal{M}_P$ is wrong or uncertain. Samples selected could be samples where $\mathcal{M}_P$ is wrong in any of detection, position, dimension, and classification. By contrast, entropy (as described in Section 2.2.1) only take class prediction into account.

The dataset is described in Section 3.1. In Section 3.2, the primary DNN, $\mathcal{M}_P$, is presented. The primary DNN is trained only on annotated images and is the DNN we try to improve using AL. In Section 3.3, we present two different secondary DNN's, $\mathcal{M}_S$. The AL algorithm using discrepancy measure as informativeness score is introduced in Section 3.4 and the calculations of the actual discrepancy are laid out in Section 3.5 and 3.6. The development process of the discrepancy measure is explained in Section 3.7. Lastly, two different ways of selecting samples based on the discrepancy are described in Section 3.8.

3.1 Dataset

The dataset we use is data provided by Zenuity. The primary motivation behind the choice of data is to make the thesis relevant to the company. Another benefit is that $\mathcal{M}_P$ already is compatible with the dataset, i.e. no modifications for $\mathcal{M}_P$ are needed. Also, the dataset is large which makes it suitable to use for AL experiments.

The dataset $\mathcal{D}$ is a set of N samples,

$$\mathcal{D} = \{\mathbf{X}_1, ..., \mathbf{X}_N\} \tag{3.1}$$

where each sample

$$\mathbf{X}_i = (\mathbf{p}, \mathbf{x}, A) \tag{3.2}$$

consists of a point cloud $\mathbf{p}$, image $\mathbf{x}$, and a set of annotations A. All coordinates in A are defined in the metric system with the ego vehicle as the origin. For each sample, $\mathbf{p}$ and $\mathbf{x}$ are captured approximately at the same time. The image $\mathbf{x}$ is captured from one of the cameras mounted around the ego vehicle. The annotations in A consists of classified 2D and 3D bounding boxes. There are ten different classes, where background is one of them. Since the point clouds and the images are captured at the same moment, it is possible to train one model with images and one model with point clouds with the same annotations.

3.2 Primary Object Detection Network

As the primary DNN, we will use a DNN based on ResNet [9] that is provided by Zenuity. For simplicity, we call the DNN $\mathcal{M}_P$ where P stands for primary. The primary DNN $\mathcal{M}_P$ is trained solely on annotated images and outputs both predicted 2D and 3D bounding boxes.

The 2D boxes are defined on the format $(x, y, w, h)^T$ where $(x, y)^T$ defines the center point (in pixels) and $(w, h)^T$ defines the dimension (width and height in pixels) of the box. The 3D boxes are defined on the format $(x, y, z, w, h, l, r)^T$ where $l = (x, y, z)^T$ defines the location (center point), $d = (w, h, l)^T$ defines the dimension (width, height and length in meters) and r defines the rotation around the y-axis (in radians), also referred to as *yaw*. Rotation around x-axis and rotation around z-axis are assumed to be negligible. An illustration of a 3D bounding box can be seen in Figure 3.1.

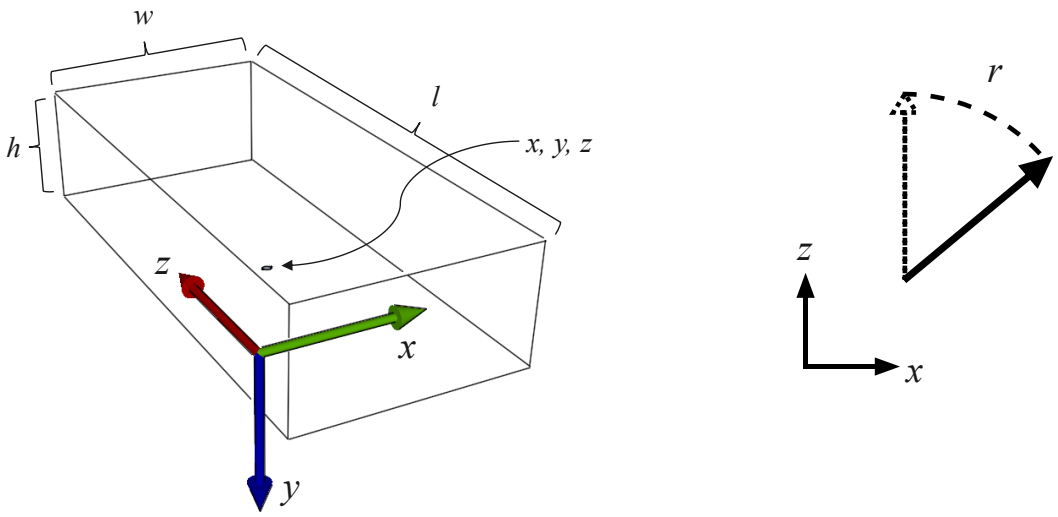


Figure 3.1: An illustration of a 3D bounding box with axis directions.

The coordinate system is in meters and the position of the camera is used as the origin. In the camera's coordinate system, the x-axis is pointing to the right, y-axis is pointing down and z-axis is pointing forward (see Figure 3.1). For example, an object located at $(x, y, z) = (10, -1, 10)$ means that it is 10 meters to the right, 1 meter below, and 10 meters in front of the camera. The rotation is defined relative to the ego vehicle. To exemplify, $r = 0$ means the object is facing in the same direction as the ego vehicle, $r = \frac{\pi}{2}$ means the object is facing to the right and $r = \pi$ means the object is facing in the opposite direction of the ego vehicle.

3.3 Secondary Object Detection Network

The secondary DNN should in principle be a DNN with high performance in 3DOD. A DNN with point clouds as input is preferable to get high performance on the 3D parameters, as described in Section 2.3. We call the secondary DNN $\mathcal{M}_S$ where S stands for secondary. Two different versions of $\mathcal{M}_S$ are used to evaluate the performance of the discrepancy measure. The two DNNs are described in the following subsections.

3.3.1 Simulated High Performance Network

To mimic a DNN with high performance, we use the annotations as if they were detections from $\mathcal{M}_S$. Using the annotations to select samples for human annotation is of course not possible in practice. The annotations are not available before they are created by a human. However, the results can still show that the discrepancy measure is worth using if a high-performance DNN is used as $\mathcal{M}_S$. We call the simulated high-performance DNN $\mathcal{M}_A$ where A stands for annotation.

3.3.2 Camera-LiDAR Fusion Network

To evaluate the practical potential of the discrepancy measure, a real Camera-LiDAR fusion DNN is used. The DNN is based on a DNN developed in a previous thesis at Zenuity [20].

The DNN is a fusion DNN which in this case means that it is trained on both images and point clouds. We call the DNN $\mathcal{M}_F$ where F stands for fusion. $\mathcal{M}_F$ is developed to be robust against sensor degradation, i.e. if one of the sensors (camera or LiDAR) stops working, the DNN should still be able to perform object detections. In other words, $\mathcal{M}_F$ is functioning using solely point clouds as input which gives an option to investigate if the fusion part, i.e. using both images and point clouds, is important for calculating discrepancy in AL. The architecture of $\mathcal{M}_F$ is explained in Section A.2 and modifications to $\mathcal{M}_F$ are explained in Section A.3.

3.4 Modified Active Learning Algorithm

We base our experiments on the pool-based AL algorithm described in 2.2. As mentioned in 2.2, the budget is denoted b, the dataset with annotations is denoted $\mathcal{A}$ and the dataset without annotations is denoted $\mathcal{N}$. We also add a new annotated set $\mathcal{V}$ which is our *validation* set. Apart from using $\mathcal{M}_P$ as $\mathcal{M}$, a few other modifications are done. These modifications are underlined in the following algorithm.

1. Train $\mathcal{M}_P$ and $\mathcal{M}_S$ on $\mathcal{A}$.
2. Evaluate $\mathcal{M}_P$ on $\mathcal{V}$.
3. Apply $\mathcal{M}_P$ and $\mathcal{M}_S$ on $\mathcal{N}$ and store the output.
4. Score the output from step 3 using a discrepancy measure between the detections of $\mathcal{M}_P$ and $\mathcal{M}_S$.
5. Pick the b samples from $\mathcal{N}$ according to some selection strategy based on discrepancy and send for annotation.
6. Add the annotated data to $\mathcal{A}$ and remove it from $\mathcal{N}$.
7. Repeat from step 1 .

First, we train not only $\mathcal{M}_P$ on $\mathcal{A}$, but also $\mathcal{M}_S$. After each training, we evaluate the performance of $\mathcal{M}_P$ on $\mathcal{V}$ for comparison between *AL iterations*. An AL iteration is one iteration of steps 1-6. In step 4 we use our discrepancy measure as informativeness score. Lastly, we select samples based on discrepancy using some selection strategy.

3.5 Main Approach

Two different approaches to quantifying discrepancy are tested. We call them Main Approach (MA) and Alternative Approach (AA). Both approaches need to

1. Match detections from $\mathcal{M}_P$ and $\mathcal{M}_S$ (as mentioned in Section 2.5).
2. Quantify the discrepancy between the matched detections.
3. Include unmatched detections in the measure.

This section describes in detail how MA matches detections and calculates total discrepancy for a sample. AA is heavily dependent on MA, which is why it is important to first read and understand this section before reading Section 3.6.

3.5.1 Matching

Let $\mathcal{X}_P = \{X_{P,1}, ..., X_{P,n}\}$ and $\mathcal{X}_S = \{X_{S,1}, ..., X_{S,k}\}$ be two sets of detections for sample $\mathbf{X}$ coming from $\mathcal{M}_P$ and $\mathcal{M}_S$, respectively. Before using the Hungarian method to find the best matchings, some detections will be disregarded. Since $\mathcal{M}_S$ is supposed to detect objects in point clouds, there could be detections that will not be visible in the corresponding image. However, $\mathcal{M}_P$ will only use images and will never be able to detect anything that is not in the image. Therefore, any detection in $\mathcal{X}_S$ that does not project into the image plane will be left out.

We will now define three matrices, $M_{\text{IoU}}, M_{\text{obj}}, M_{\text{depth}}$ that are to be used to calculate the matrix M used by the Hungarian method to solve the 2DA. We define

$$M_{\text{IoU}} = \begin{pmatrix} \text{IoU}_{\text{2D}}(X_{P,1}, X_{S,1}) & \dots & \text{IoU}_{\text{2D}}(X_{P,1}, X_{S,k}) \\ \vdots & \ddots & \vdots \\ \text{IoU}_{\text{2D}}(X_{P,n}, X_{S,1}) & \dots & \text{IoU}_{\text{2D}}(X_{P,n}, X_{S,k}) \end{pmatrix} \tag{3.3}$$

to be the $n \times k$ *IoU matrix* that contains the 2D IoU scores between all possible combinations of the detections in $\mathcal{X}_P$ and $\mathcal{X}_S$. If $\mathcal{M}_S$ does not output 2D boxes, the 2D box is created by bounding the 3D box that is projected into the image plane.

Furthermore, we define p_0 to be the probability of a detection being background, i.e. the probability that a detection is neither any of the 9 other classes. The *objectness probability* is defined as $1 - p_0$, i.e. the probability of the detection being any of the 9 other classes. Then M_{obj} is the $n \times k$ *objectness matrix* between $\mathcal{X}_P$ and $\mathcal{X}_S$ and is defined

$$M_{\text{obj}} = \begin{pmatrix} 1 - p_{0,P,1} \\ \vdots \\ 1 - p_{0,P,n} \end{pmatrix} \begin{pmatrix} 1 - p_{0,S,1} & \dots & 1 - p_{0,S,k} \end{pmatrix} \tag{3.4}$$

i.e. the element at position (i, j) is the product of the objectness probabilities of detection i from $\mathcal{X}_P$ and detection j from $\mathcal{X}_S$.

Furthermore, we define the *relative depth difference* between detections $X_{P,i}$ and $X_{S,j}$ to be

$$\Delta_{ij}^{z} = \frac{\min(z_{P,i}, z_{S,j})}{\max(z_{P,i}, z_{S,j})} \tag{3.5}$$

where $z_{P,i}$ and $z_{S,j}$ is the center point z-coordinate of detections $X_{P,i}$ and $X_{S,j}$, respectively. M_{depth} is then defined

$$M_{\text{depth}} = \begin{pmatrix} \Delta_{11}^{z} & \dots & \Delta_{1k}^{z} \\ \vdots & \ddots & \vdots \\ \Delta_{n1}^{z} & \dots & \Delta_{nk}^{z} \end{pmatrix}. \tag{3.6}$$

The three matrices will now be multiplied together using the Hadamard product $\circ$ (element-wise multiplication) to form a *score matrix*

$$M = M_{\text{IoU}} \circ M_{\text{obj}} \circ M_{\text{depth}}. \tag{3.7}$$

The purpose of the IoU matrix M_{IoU} is to have matched detections close to each other in position, which is the most crucial part. The objectness matrix M_{obj} aims to ensure that both detections in a match have high confidence of detecting a real-world object, otherwise, there is a risk of matching a false positive. Finally, M_{depth} prevents that the difference between detections in a match is unrealistically large in depth, which is possible if only matching with respect to 2D IoU.

The final step before solving the 2DA using the Hungarian method on M is to filter out any matchings that are not feasible. This is determined using two thresholds τ_{IoU} and τ_{depth}. If an element at position (i, j) in M_{IoU} is smaller than τ_{IoU}, the corresponding element m_{ij} in M will be set to 0. The same applies to M_{depth} using τ_{depth}.

Before running the Hungarian method, dummy rows or columns will be added as described in Section 2.5.1 if M is not a square matrix. Let $n' = \max(n, k)$. The 2DA is originally formulated to minimize the total cost of the problem (see equation (2.9)). In this case, we want to maximize the total matching score, i.e.

$$\text{maximize} \quad \sum_{i=1}^{n'}\sum_{j=1}^{n'} y_{ij}m_{ij} \tag{3.8a}$$

$$\text{subject to} \quad \sum_{i=1}^{n'} y_{ij} = 1, \quad \forall j \in \{1, ..., n'\}, \tag{3.8b}$$

$$\sum_{j=1}^{n'} y_{ij} = 1, \quad \forall i \in \{1, ..., n'\}, \tag{3.8c}$$

$$y_{ij} \in \{0, 1\}, \quad \forall i, j \in \{1, ..., n'\} \tag{3.8d}$$

where $y_{ij} = 1$ if the ith object in $\mathcal{X}_P$ is assigned the jth object in $\mathcal{X}_S$. To be able to use the Hungarian method for finding the best matchings we need to convert the problem into a minimization problem. The objective function in (3.8a) is converted to

$$\text{minimize} \quad \sum_{i=1}^{n'}\sum_{j=1}^{n'} y_{ij}\left((\max_{(i,j)} m_{ij}) - m_{ij}\right). \tag{3.9}$$

Now, the previously largest value in M has become the smallest value and the previously smallest values has become the largest.

From the Hungarian method, we receive pairs of indices of the form (i, j) where i is the index of a detection from $\mathcal{X}_P$ and j the index of a detection from $\mathcal{X}_S$. Any pair where $i > n$ or $j > k$ is a match coming from dummy rows or dummy columns and will be discarded. Furthermore, three things need to hold for a match (i, j) to be valid:

1. The element m_{ij} in M needs to be > 0.
2. The element at position (i, j) in M_{IoU} needs to be $\geq \tau_{\text{IoU}}$.
3. The element at position (i, j) in M_{depth} needs to be $\geq \tau_{\text{depth}}$.

Any match that does not fulfill these conditions will be discarded. The matchings that are not discarded are put in a set

$$\mathcal{X}^{\text{match}} = \{(X_{P,a}, X_{S,b}), ..., (X_{P,c}, X_{S,d})\} \tag{3.10}$$

where a and b are indices of detections from $\mathcal{X}_P$ and $\mathcal{X}_S$, respectively, that are matched to each other.

Since $\mathcal{M}_S$ uses point clouds that have a finite range, there might be detections from $\mathcal{M}_P$ that will never be matched to anything from $\mathcal{M}_S$ due to their position being beyond the range of the point cloud. These unmatched detections from $\mathcal{M}_S$ would then lead to a non-constructive discrepancy which is why they are left out. This is done by filtering out any unmatched detections from $\mathcal{M}_P$ that have a z-coordinate further away than δ_z, where δ_z is set to be the maximum range of the LiDAR sensor. The number of detections from $\mathcal{X}_P$ and $\mathcal{X}_S$ that are among the discarded matchings after the filtering are denoted μ_P and μ_S, respectively.

The number of detections μ_P is assumed to be μ_P objects which $\mathcal{M}_P$ detects but $\mathcal{M}_S$ does not and μ_S is assumed to be μ_S objects which $\mathcal{M}_S$ detects but $\mathcal{M}_P$ does

not. Both μ_P and μ_S are included in the discrepancy measure where the assumption is that μ_P is the number of false detections and μ_S is the number of detections that $\mathcal{M}_P$ missed. By letting μ_P affect the discrepancy measure, we intend to select samples that increase precision performance. Similarly, we intend to select samples that increase recall performance by letting μ_S affect the discrepancy measure. We refer to Section 3.5.3 for details regarding how the discrepancy measure is affected by μ_P and μ_S.

3.5.2 Discrepancy between a Pair of Matched Detections

We have two detections, X_P and X_S coming from $\mathcal{M}_P$ and $\mathcal{M}_S$, respectively. The discrepancy in attributes, i.e. the position, dimension, and rotation, are taken into account by calculating the absolute difference between the attributes of X_P and X_S. For a detection X we define the attributes of X as

$$X^{\text{attr}} = \begin{pmatrix} x \\ y \\ z \\ h \\ w \\ l \\ r \end{pmatrix} \tag{3.11}$$

where x, y, z is the center point, h, w, l the dimension and r the rotation around the y-axis (yaw) of X (as described in Section 3.2). The rotation is modified as $r := r \mod \pi$ such that $r \geq 0$. The absolute differences between the attributes of X_P and X_S are then

$$|X_P^{\text{attr}} - X_S^{\text{attr}}| = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta h \\ \Delta w \\ \Delta l \\ \Delta r \end{pmatrix}. \tag{3.12}$$

To avoid collisions, objects closer to the ego vehicle are often more important to accurately predict than objects further away. Therefore, we define relative depth difference as $\Delta z_{\text{rel}} = \Delta z / \max(z_P, z_S)$. Since the output from $\mathcal{M}_S$ could be invariant to heading direction, as in $\mathcal{M}_F$ (see (A.8)), the rotation difference need to be invariant as well. We define the invariant rotation difference as $\Delta r_{\text{inv}} = \min(\Delta r, \pi - \Delta r)$. The maximum difference in rotation is therefore $\pi/2$, as seen in Figure 3.2. We define the difference between the attributes as

$$\Delta^{\text{attr}} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z_{\text{rel}} \\ \Delta h \\ \Delta w \\ \Delta l \\ \Delta r_{\text{inv}} \end{pmatrix}. \tag{3.13}$$

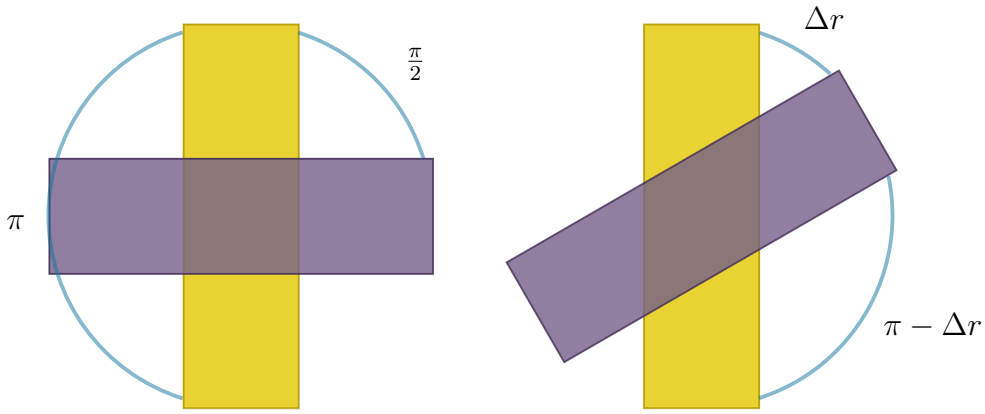


Figure 3.2: Illustration of rotation difference between two bounding boxes. The difference is largest when the bounding boxes are orthogonal.

The different components of Δ^{attr} differ in magnitude which later will make it hard to combine them into a single measure. Therefore, we normalize Δ^{attr} using a vector

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_x \\ \beta_y \\ \beta_z \\ \beta_h \\ \beta_w \\ \beta_l \\ \beta_r \end{pmatrix} \tag{3.14}$$

containing maximum difference values that are set based on observations in the dataset $\mathcal{D}$. The normalized vector of attribute differences is the *attribute discrepancy* vector

$$\mathbf{d}^{\text{attr}} = \Delta^{\text{attr}} \oslash \boldsymbol{\beta} = \begin{pmatrix} d_x \\ d_y \\ d_z \\ d_h \\ d_w \\ d_l \\ d_r \end{pmatrix} \tag{3.15}$$

where $\oslash$ is the Hadamard (element-wise) division operator.

Furthermore, to include the discrepancy in class prediction, we calculate the L1-norm of the difference between the probability vectors of X_P and X_S. The probability vector of a detection X is

$$X^{\text{prob}} = \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_9 \end{pmatrix} \tag{3.16}$$

where p_0 is the probability of X being background and $p_1, ..., p_9$ are probabilities for the 9 different classes. The L1-norm of the difference (the *probability discrepancy*) is

$$d^{\text{prob}} = ||X_P^{\text{prob}} - X_S^{\text{prob}}||. \tag{3.17}$$

Combined into one column vector, the attribute discrepancies and the probability discrepancy is denoted

$$\mathbf{d} = \begin{pmatrix} \mathbf{d}^{\text{attr}} \\ d^{\text{prob}} \end{pmatrix} = \begin{pmatrix} d_x \\ d_y \\ d_z \\ d_h \\ d_w \\ d_l \\ d_r \\ d_{\text{prob}} \end{pmatrix}. \tag{3.18}$$

3.5.3 Discrepancy for a Sample

Let

$$k' = |\mathcal{X}^{\text{match}}| \tag{3.19}$$

be the total number of pairs with matched detections for sample $\mathbf{X}$. Then, let

$$D = \begin{pmatrix} \mathbf{d}_1 & ... & \mathbf{d}_{k'} \end{pmatrix} \tag{3.20}$$

be the matrix containing all column vector discrepancies (3.18) for the matched detections of $\mathbf{X}$. We want to avoid only selecting samples that contain many objects since they take more time to annotate. Samples with many objects are therefore often more expensive to annotate than samples with fewer objects. To prevent over-selection of samples where k' is large, we take the row-wise average of D. If instead, a row-wise sum is used, the discrepancy of a sample will be directly correlated to the number of objects. Then, samples that are expensive to annotate will be selected first. Thereby, let row_{avg} be the function that calculates the column vector containing the row-wise average values of a matrix. Then, let

$$\bar{\mathbf{d}} = \text{row}_{\text{avg}}(D). \tag{3.21}$$

Now, let

$$\mathbf{w} = \begin{pmatrix} w_x \\ w_y \\ w_z \\ w_h \\ w_w \\ w_l \\ w_r \\ w_{\text{prob}} \\ w_P \\ w_S \end{pmatrix} \tag{3.22}$$

be a vector of weights. The *weighted discrepancy* vector is then calculated as

$$\mathbf{d}_w = \mathbf{w} \circ \begin{pmatrix} \bar{\mathbf{d}} \\ f(\mu_P) \\ f(\mu_S) \end{pmatrix} \tag{3.23}$$

where f is a function determining the discrepancy for unmatched objects. We consider $\mathbf{w}$ and f to be a parameters which can be set to change the functionality of the algorithm. The parameter settings we use in our experiments are explained in Section 4.4.

The final step is now to combine all the components into a single scalar value, which is done using a squared L2-norm, i.e.

$$d = ||\mathbf{d}_w||_2^2 \tag{3.24}$$

where d denotes the total discrepancy value for sample $\mathbf{X}$.

3.5.4 Main Approach Examples

Figure 3.3 shows an example of a sample with low discrepancy value and Figure 3.4 an example of a sample with a higher discrepancy value. In Appendix A.1 the matrices M_{IoU}, M_{obj} and M_{depth} corresponding to the problem instance in Figure 3.3 are shown as well as the matrix M used for solving the 2DA with the Hungarian method.

3.6 Alternative Approach

AA is based on the assumption that a good match between detections is a match with low discrepancy. Instead of matching the detections using $M_{\text{IoU}} \circ M_{\text{obj}} \circ M_{\text{depth}}$, as in MA, the detections are essentially matched using the discrepancy measure explained in Section 3.5.2. The Hungarian method is used to match detections such that the total discrepancy is minimized.

Let $\mathcal{X}_P = \{X_{P,1}, ..., X_{P,n}\}$ and $\mathcal{X}_S = \{X_{S,1}, ..., X_{S,k}\}$ be two sets of detections for sample $\mathbf{X}$ coming from $\mathcal{M}_P$ and $\mathcal{M}_S$, respectively. Then

$$d_{ij} = ||\mathbf{w}_{\text{matched}} \circ \mathbf{d}_{ij}||_2^2 \tag{3.25}$$

is the discrepancy between detections $X_{P,i}$ and $X_{S,j}$ where $\mathbf{d}_{ij}$ is the attribute and probability discrepancies between $X_{P,i}$ and $X_{S,j}$ according to (3.18). $\mathbf{w}_{\text{matched}}$ is defined as $\mathbf{w}$ from (3.22) without the last two elements w_P and w_S. We then define the *discrepancy matrix* to be

$$M_{\text{disc}} = \begin{pmatrix} d_{11} & \dots & d_{1k} \\ \vdots & \ddots & \vdots \\ d_{n1} & \dots & d_{nk} \end{pmatrix}. \tag{3.26}$$

Figure 3.3: Example of sample where the discrepancy is small. There is only a small difference in depth between the matched detections and the largest difference are detections far away from the ego vehicle, meaning small discrepancy since the depth difference is relative.

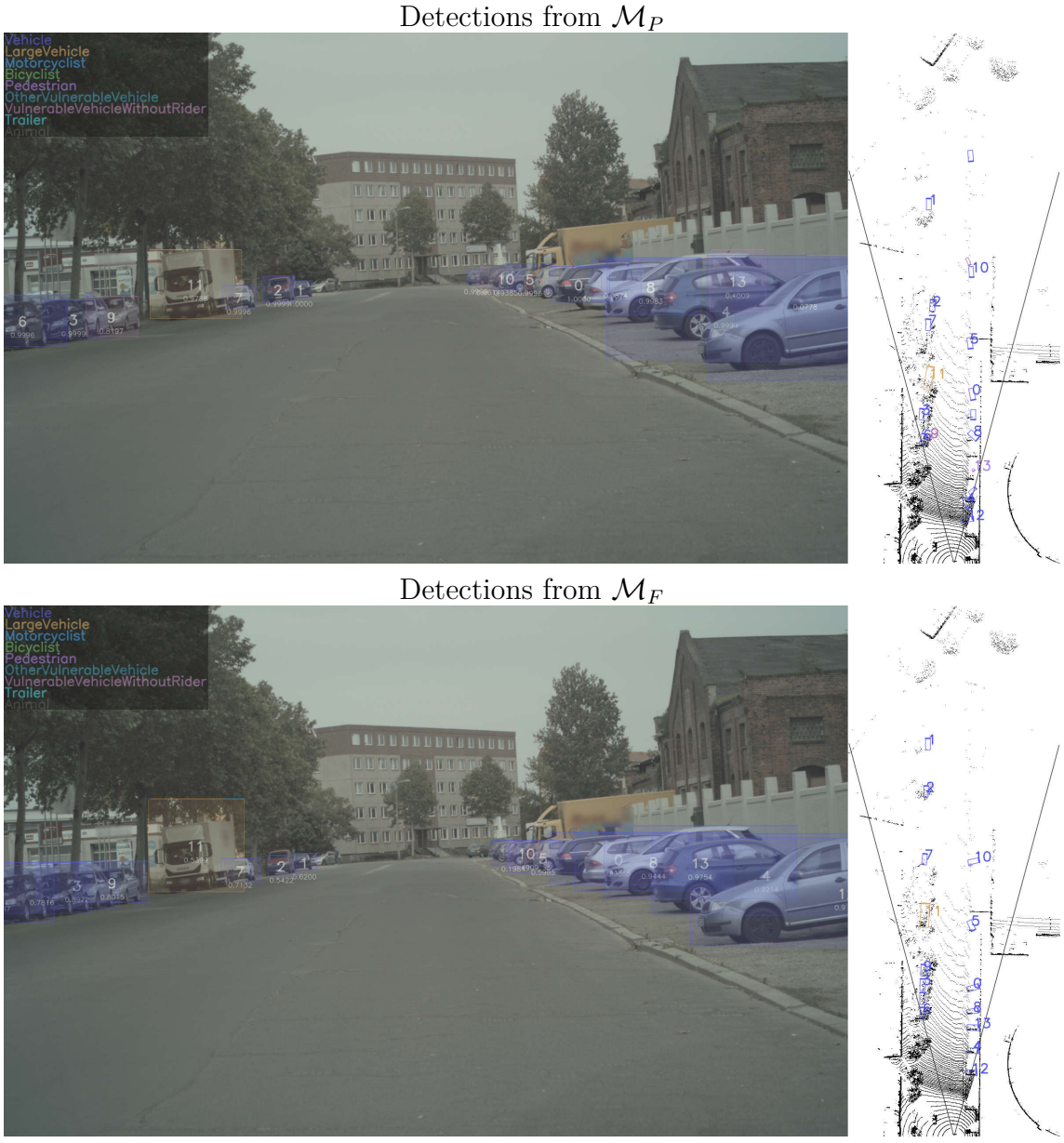


Figure 3.4: Example of sample where the discrepancy is large. $\mathcal{M}_F$ is more accurate in position than $\mathcal{M}_P$ as seen in the BEV point cloud to the right.

For solving the 2DA using the Hungarian method, we then construct M by

$$M = M_{\text{disc}} \odot (1 - M_{\text{IoU}}) \tag{3.27}$$

where M_{IoU} is the IoU matrix as described in 3.5.1. Since we want to minimze the total discrepancy and since we use $1 - M_{\text{IoU}}$ instead of M_{IoU}, the 2DA on M is a minimization problem. Similar to Section 3.5.1, if an element at position (i, j) in M_{IoU} or M_{depth} is smaller than τ_{IoU} or τ_{depth}, respectively, the corresponding element m_{ij} in M will be changed. Since M now corresponds to a minimization problem, the values will not be set to 0 as in Section 3.5.1 but instead a very large value, e.g. 99999. Note that M is changed for elements in M_{depth} that are smaller than τ_{depth}

even if M_{depth} is not used to calculate M. Dummy rows and columns are added if necessary before using the Hungarian method to solve the 2DA on M.

The returned pairs of indices from the Hungarian method are filtered in the same way as in 3.5.1. Let E_{matched} be the set containing the filtered and matched indices for the detections from $\mathcal{X}_P$ and $\mathcal{X}_S$. That is, $(i, j) \in E_{\text{matched}}$ means detection with index i in $\mathcal{X}_P$ is matched to index j in $\mathcal{X}_S$. The discrepancy of matched detections, d_{matched}, is then

$$d_{\text{matched}} = \frac{\sum_{(i,j)\in E_{\text{matched}}} m_{ij}}{k'} \tag{3.28}$$

where $k' = |E_{\text{matched}}|$ and m_{ij} is the element at position (i, j) in M.

The discrepancy of unmatched detections, $d_{\text{unmatched}}$ is

$$d_{\text{unmatched}} = || \begin{pmatrix} w_P \\ w_S \end{pmatrix} \circ \begin{pmatrix} \mu_P \\ \mu_S \end{pmatrix} ||_2^2. \tag{3.29}$$

The total discrepancy for sample $\mathbf{X}$ is then the sum of d_{matched} and $d_{\text{unmatched}}$

$$d = d_{\text{matched}} + d_{\text{unmatched}}. \tag{3.30}$$

3.7 Discrepancy Measure Development Process

Section 3.5 and 3.6 describe two approaches to quantifying discrepancy. Both are built around the same central measure of discrepancy. The different components of the discrepancy measure, as well as their parameters, are chosen through a development process relying heavily on subjective decisions.

The process involves a small set of ($\sim$ 400) randomly selected samples and for each sample, two sets of cached detections coming from $\mathcal{M}_P$ and $\mathcal{M}_F$, respectively. Then, the samples and their detections are all processed using MA or AA (depending on what is being developed) and each sample is assigned a discrepancy. By manually looking at the images (example images are shown in Figure 3.5) and subjectively relating their discrepancy to how valuable they are believed to be for $\mathcal{M}_P$ to train on, design choices are made and parameter values are set. The goal is for the discrepancy measure to assign a large discrepancy to samples where $\mathcal{M}_P$ is believed to benefit from learning.

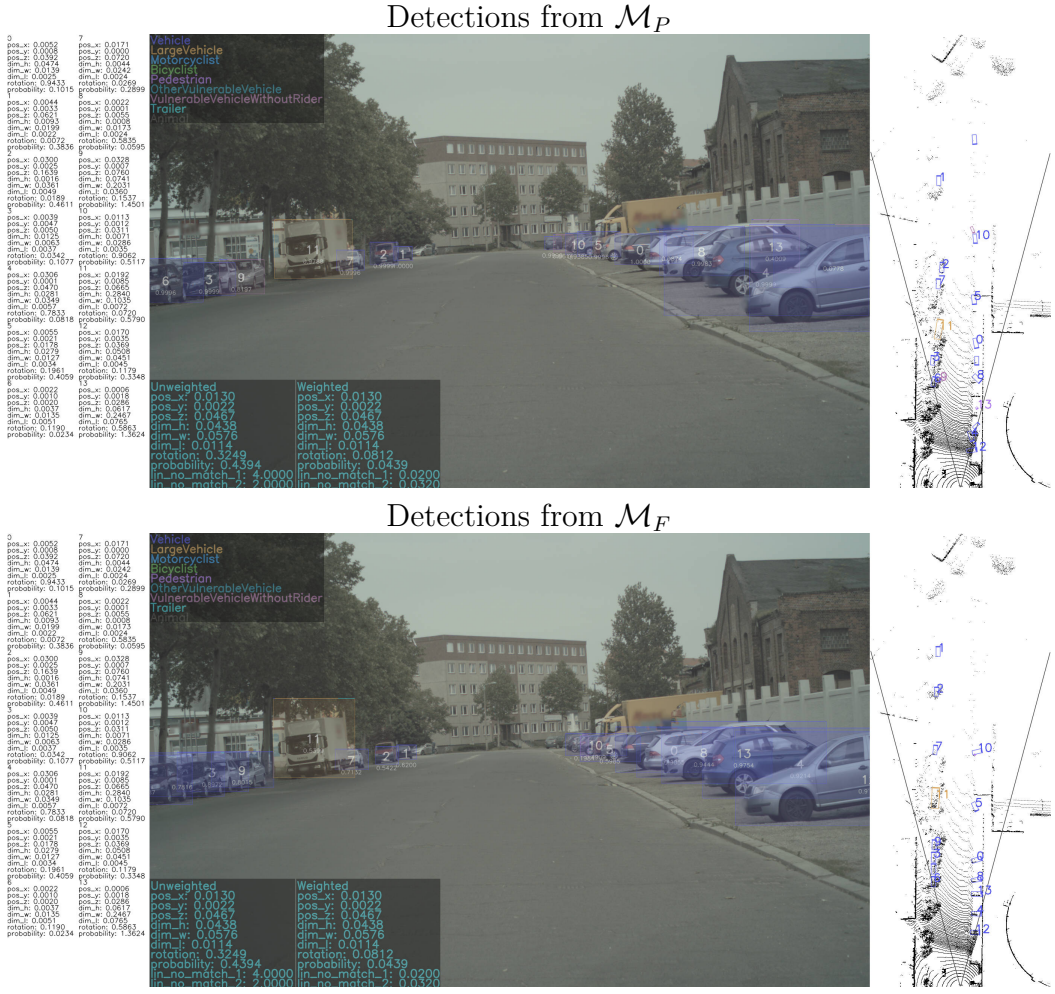


Figure 3.5: Example images used when developing the discrepancy measure. Discrepancy values for each pair of matched detections are shown to the left. The dark boxes in the bottom left of the image show unweighted and weighted mean discrepancy values.

3.8 Selection Strategy

After the discrepancy for each sample in the non-annotated data set is calculated, the samples to send for annotation are selected. We compare two different selection strategies described in the following subsections.

3.8.1 Standard Selection

First, $\mathcal{N}$ is sorted according to the discrepancy. The b samples from $\mathcal{N}$ with largest discrepancy are selected for annotation, as in Section 2.2. We refer to this selection strategy as Standard Selection (SS).

By only considering the largest discrepancy when selecting, the resulting selected set of samples is not necessarily diverse. For example, suppose we evaluate on a dataset containing solely of front camera images. If the selected set of samples consists of solely back camera images, the DNN will most likely perform poorly. However, selecting only front camera images, in this case, is not optimal either, as there could still be valuable information in the other camera angles. A more diverse set of samples can be achieved by selecting samples from each camera angle, as described in the next subsection.

3.8.2 Camera Angle Selection

As in SS, Camera Angle Selection sorts $\mathcal{N}$ according to discrepancy. Still, b samples are selected for annotation but there are restrictions. Only a fraction of b can be selected for each camera angle and the fractions are fixed and determined before the AL run begins. For example, suppose b must contain 50% front camera images and 50% back camera images. The b samples will consist of $b/2$ images with the largest discrepancy out of the front camera images and $b/2$ images with the largest discrepancy out of the back camera images. We refer to this selection strategy as Camera Angle Selection (CAS).

4

Experiment Settings

This chapter explains in detail how the experiments are set up. We select subsets of the whole dataset to speed up experiments and this is explained in Section 4.1. Furthermore, we describe how the method is evaluated and compared to other methods in Section 4.2. We give details on the setting of the AL algorithm in 4.3 and the parameter setting for the discrepancy measure is explained in Section 4.4.

4.1 Experiment Data Selection

Our experiments are done on $\mathcal{D}$ (see Section 3.1) and contains around 80000 annotated samples. We use a 90/10 train/validation split when creating the training set $\mathcal{D}^T$ and the validation set $\mathcal{D}^V$. The split is done based on the date when the data is gathered instead of splitting randomly to avoid getting too similar images in training and validation. The training set $\mathcal{D}^T$ contains images taken from several camera angles. However, $\mathcal{D}^V$ contains images taken from only front cameras. The reason is that front cameras are more important than any other camera angle in a self-driving car.[1]

All data in $\mathcal{D}$ have annotations. For the experiments, some of that data is treated as if it does not have annotations, as described in Section 2.2.2. In that case, the annotations are simply hidden or not used by the algorithm. The annotations are then revealed upon the AL algorithms request.

All experiments are initialized with $\mathcal{N}$ to be a random subset of $\mathcal{D}^T$ of size 30000. The reduction in size speeds up the experiments and makes it easier to iterate different methods. A random subset of $\mathcal{N}$ of size 3000 is then moved to $\mathcal{A}$ (and the annotations are revealed) as an initial training set for the AL algorithm. Furthermore, 1000 randomly selected samples from $\mathcal{D}^V$ are moved to a set $\mathcal{V}$ which is used in the experiments as the validation set. The random selections that are done to select the initial data for the non-annotated dataset $\mathcal{N}$, annotated dataset $\mathcal{A}$, and validation dataset $\mathcal{V}$ are provided with a random seed. This enables us to reproduce experiments with the same initial data sets by using identical random seeds. It also enables cross-evaluation of our methods with different initial data sets using different random seeds.

[1] Just consider the fraction of the time you are looking forward while driving.

4.2 Evaluation

We evaluate our method by measuring Key Performance Indicator (KPI) gains on $\mathcal{M}_P$ when evaluating on $\mathcal{V}$. Our main KPIs are AP and F1 score. AP and F1 score are both dependent on IoU and IoU between 3D bounding boxes are often close to zero since the position is not always accurately predicted in $\mathcal{M}_P$. Therefore, both KPIs are calculated using IoU between 2D bounding boxes.

AP is calculated as in the VOC2007 challenge, i.e. (2.26), but with a step size of 101, meaning interpolated precision value for $r = 0, 0.01, 0.02, ..., 0.99, 1$. The confidence is calculated as $1 - p_0$, meaning any class that is not background is taken into account.

The F1 score is calculated for 101 different confidence thresholds. The confidence thresholds are $0, 0.01, 0.02, ..., 0.99, 1$. A detection is only counted as long as it is above the threshold. The maximum F1 score calculated for all the 101 thresholds is presented.

Since AP and F1 score are insensitive to 3D position, it is of interest to see if the performance in position improves. As mentioned in Section 2.3, predicting an object's position in a point cloud is less difficult than in an image. Therefore, $\mathcal{M}_S$ most likely predicts position for a detection more accurately than $\mathcal{M}_P$ and a large discrepancy in position could expose samples where $\mathcal{M}_P$ is inaccurate. We measure Mean Absolute Error (MAE) in center point coordinates between detections and the annotated objects. As objects closer to the ego vehicle are more important to accurately predict, we calculate relative position error. For a given detection X and an annotation Y having positions

$$X^{\text{pos}} = \begin{pmatrix} x^X \\ y^X \\ z^X \end{pmatrix}, \quad Y^{\text{pos}} = \begin{pmatrix} x^Y \\ y^Y \\ z^Y \end{pmatrix} \tag{4.1}$$

respectively, we have the relative position error

$$\varepsilon = \frac{||Y^{\text{pos}} - X^{\text{pos}}||_2}{||Y^{\text{pos}}||_2} \tag{4.2}$$

where the denominator is the distance from the ego vehicle to the annotated object since the ego vehicle is the origin (see Section 3.1). We calculate MAE in position as

$$\text{MAE}_{\text{pos}} = \frac{1}{N} \sum_{i=1}^{N} \varepsilon_i \tag{4.3}$$

where N is the total number of true positives.

4.2.1 Baselines

We compare our method to a main baseline that is based on selecting samples randomly. The main baseline is defined as the average of 3 independent runs (but with

the same initial data as described in Section 4.1). We also compare to uncertainty sampling with entropy (see Section 2.2). Each sample is scored according to the formula

$$-\sum_{i=0}^{n}\sum_{j=0}^{9} p_{j,P,i}\log(p_{j,P,i}) \tag{4.4}$$

where $p_{j,P,i}$ denotes the probability of class j for object i. The samples with largest entropy are selected for annotation.

4.3 Active Learning Setting

The budget b is set to 3000 samples with the motivation that a sufficient number of samples need to be added in order to make a difference to the model parameters. The experiment is run for 5 AL iterations, meaning $\mathcal{M}_P$ is first trained on the initial data set $\mathcal{A}$ and continues training on 6000, 9000, 12000, 15000 and lastly 18000 samples. Each AL iteration trains for 100 epochs, including the training on the initial training set. 100 epochs are not a guarantee for total convergence. However, 100 epochs keep the time for experiments low and we believe that 100 epochs are enough to measure performance for an AL iteration. Evaluation, i.e. KPI calculation, on $\mathcal{V}$ runs every 5 epochs. For each KPI, the value for an AL iteration is the mean value of the last 5 evaluations, i.e. the last 25 epochs.

4.4 Discrepancy Measure Parameter Setting

Weights are set to define the importance of each attribute. By looking at examples like Figure 3.5 as described in Section 3.7, weights can be set to determine whether the discrepancy for a specific sample should be small or large. Furthermore, by setting an attribute weight to zero, we can investigate whether the attribute has a positive or negative effect when selecting samples.

The parameters are set to

$$\boldsymbol{\beta} = \begin{pmatrix}\beta_x\\ \beta_y\\ \beta_z\\ \beta_h\\ \beta_w\\ \beta_l\\ \beta_r\end{pmatrix} = \begin{pmatrix}80\\ 10\\ 1\\ 5\\ 5\\ 50\\ \pi/2\end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix}w_x\\ w_y\\ w_z\\ w_h\\ w_w\\ w_l\\ w_r\\ w_{prob}\\ w_P\\ w_S\end{pmatrix} = \begin{pmatrix}1\\ 0.125\\ 0.6\\ 1\\ 1\\ 1\\ 0.25\\ 0.1\\ 0.005\\ 0.016\end{pmatrix}, \quad f(\mu)=\mu, \quad \delta_z = 160. \tag{4.5}$$

An object in $\mathcal{D}$ captured in an image has its center point approximately located in a range of [-40, 40] meters in the x-coordinate and [-5, 5] meters in the y-coordinate.

Since Δz is relative and already ranges between 0 and 1, we set $\beta_z = 1$. Therefore, the maximum difference values for center point are set to $(\beta_x, \beta_y, \beta_z) = (80, 10, 1)$. For the maximum difference in dimension, $(\beta_h, \beta_w, \beta_l) = (5, 5, 50)$ is the theoretical maximum size of a large object in D, e.g. a train. The maximum difference in rotation is $\beta_r = \pi/2$ is, as explained in Section 3.5.2.

The choice of weights depends on the discrepancy between $\mathcal{M}_P$ and $\mathcal{M}_F$ detections. There is a larger difference in the z-coordinate, rotation r, and probability than the other attributes. Therefore, we set w_z, w_r and w_{prob} smaller in order to put more focus on other attributes.

Almost all the objects are located in the same height relative to the ego vehicle, which makes the y-coordinate quite easy to predict. Since the y-coordinate is not as important to improve as the other coordinates we set w_y to be small.

To have a linear growth for unmatched objects, we set $f(\mu) = \mu$. We set $w_S > w_P$ since we value recall higher than precision. Furthermore, we set $\delta_z = 160$, i.e. the maximum range of the LiDAR sensor, because $\mathcal{M}_F$ does not detect objects where $z > 160$.

We also conduct experiments using $f(\mu) = 1.1^\mu$ to investigate how an exponential function affects the performance of the method. When the exponential function is tested, weights $w_P = 0.01$ and $w_S = 0.05$ are used.

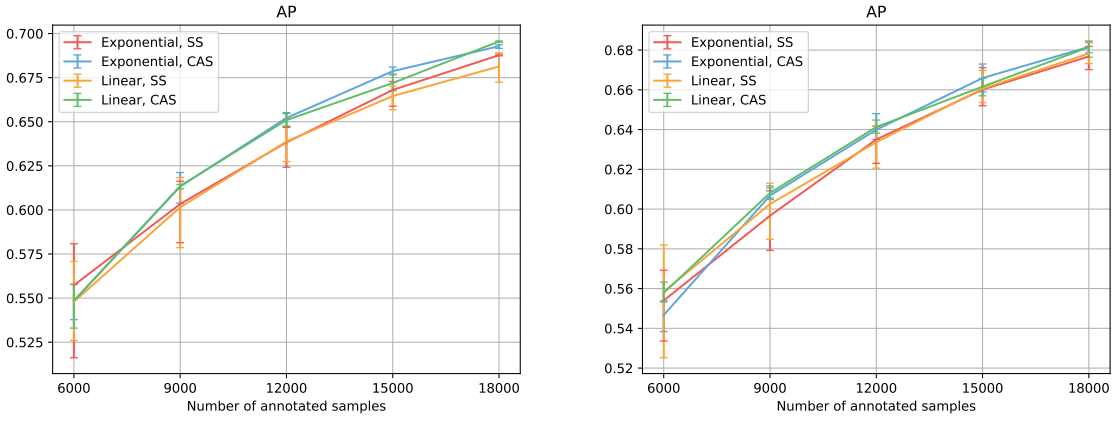
5
Results

This chapter presents our results, beginning with using annotations as outputs for $\mathcal{M}_S$ in Section 5.1 and continuing with using $\mathcal{M}_F$ in a real-case scenario in Section 5.2.

5.1 Annotation Discrepancy

As described in Section 3.3.1, the first experiments measure the discrepancy between the detections from $\mathcal{M}_P$ and $\mathcal{M}_A$, i.e. the annotations, to verify the approach before using a real DNN as $\mathcal{M}_S$. In Section 5.1.1 to 5.1.4, the Main Approach (MA) is tested and different parameter settings are compared. Then, in Section 5.1.3, MA and AA are compared. Finally, the discrepancy method is compared to the entropy and random baselines in Section 5.1.5.

Presented KPIs are AP, F1 score for pedestrians, bicyclists, and vehicles, as well as position MAE. All figures contain plots showing the results from two different random seeds for initial data selection, respectively. Each plotted line is the average of 3 independent runs with equal settings. The lines are accompanied by limits, indicating the maximum and minimum values observed for the 3 runs.



(a) Random seed 1

(b) Random seed 2

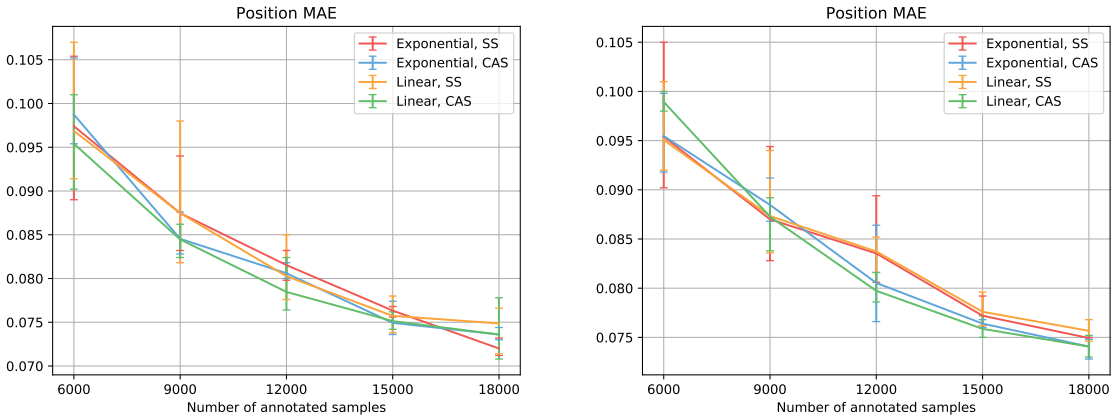
Figure 5.1: AP. Comparison between SS and CAS as well as linear and exponential growth for unmatched detections.

5.1.1 Selection Strategy

As described in Section 3.8, two different selection strategies are tested, Camera Angle Selection (CAS) and Standard Selection (SS). By looking at AP (in Figure 5.1), it is quite clear that CAS contributes to better performance than SS. Position MAE (Figure 5.2) and F1 score for vehicles (Figure 5.3) also indicate better performance using CAS than SS. Furthermore, F1 score for pedestrians (Figure 5.4) and bicyclists (Figure 5.5) fluctuate too much to give a clear picture of the differences between CAS and SS. We can, however, note that during the last 2 AL iterations, CAS is better than SS on F1 score for pedestrians.

5.1.2 Unmatched Detections

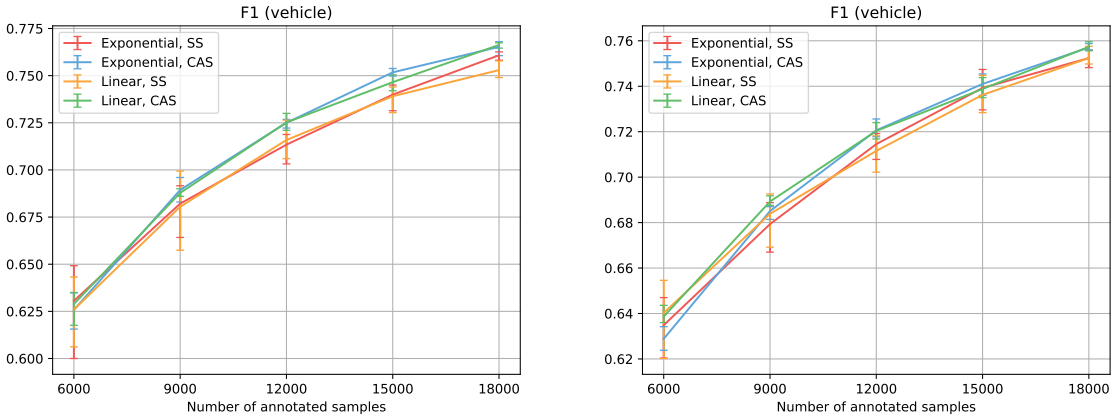
Besides comparing CAS to SS, we also compare using $f(\mu) = \mu$ to using $f(\mu) = 1.1^{\mu}$, i.e. using linear or exponential growth for the number of unmatched objects. AP (Figure 5.1) shows almost no differences between the usage of a linear or exponential f. Position MAE (Figure 5.2) shows similar results. The average values of position MAE hint that a linear f using CAS is the best approach. However, that observation is considered to be within the margin of error. We consider the exponential function to be the better choice with the note that the differences are minimal.



(a) Random seed 1

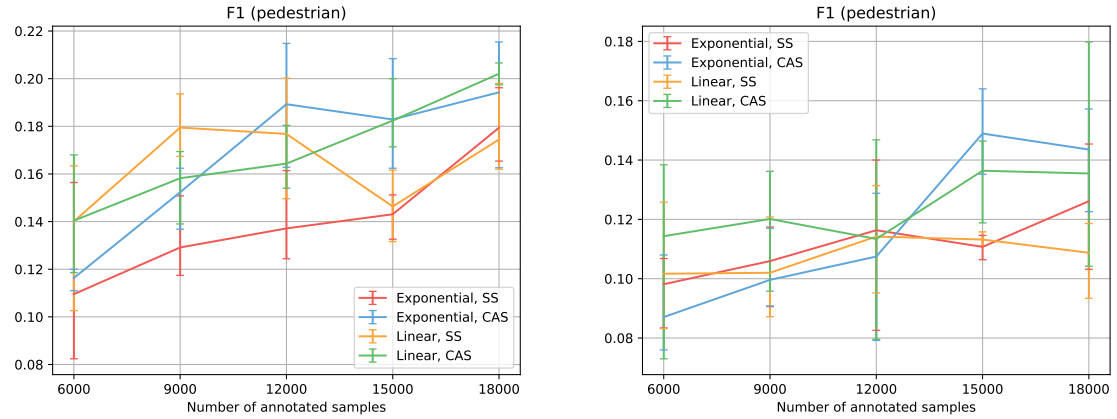
(b) Random seed 2

Figure 5.2: Position MAE. Comparison between SS and CAS as well as linear and exponential growth for unmatched detections.



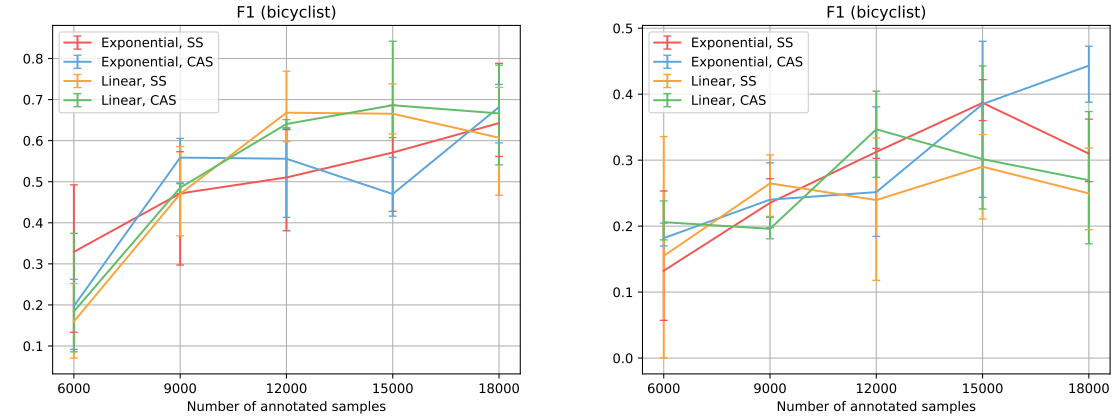
(a) Random seed 1 **(b)** Random seed 2

Figure 5.3: F1 score for vehicles. Comparison between SS and CAS as well as linear and exponential growth for unmatched detections.



(a) Random seed 1 **(b)** Random seed 2

Figure 5.4: F1 score for pedestrians. Comparison between SS and CAS as well as linear and exponential growth for unmatched detections.

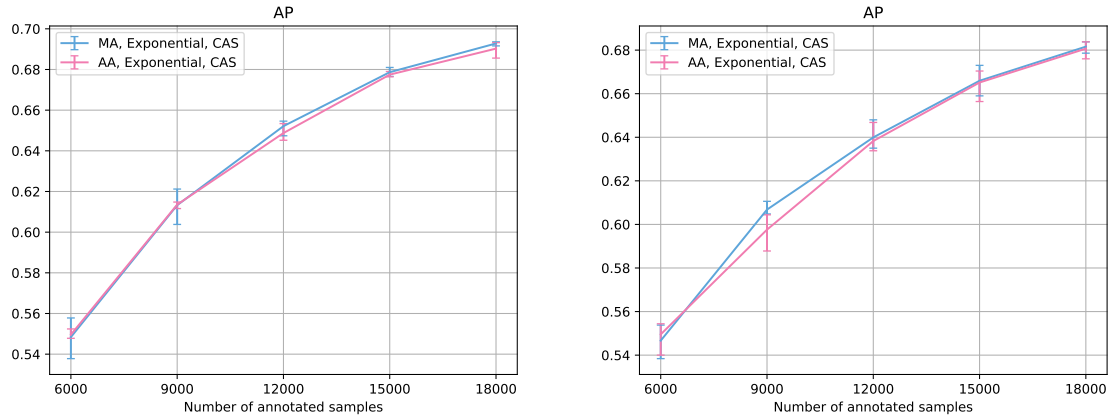


(a) Random seed 1 **(b)** Random seed 2

Figure 5.5: F1 score for bicyclists. Comparison between SS and CAS as well as linear and exponential growth for unmatched detections.

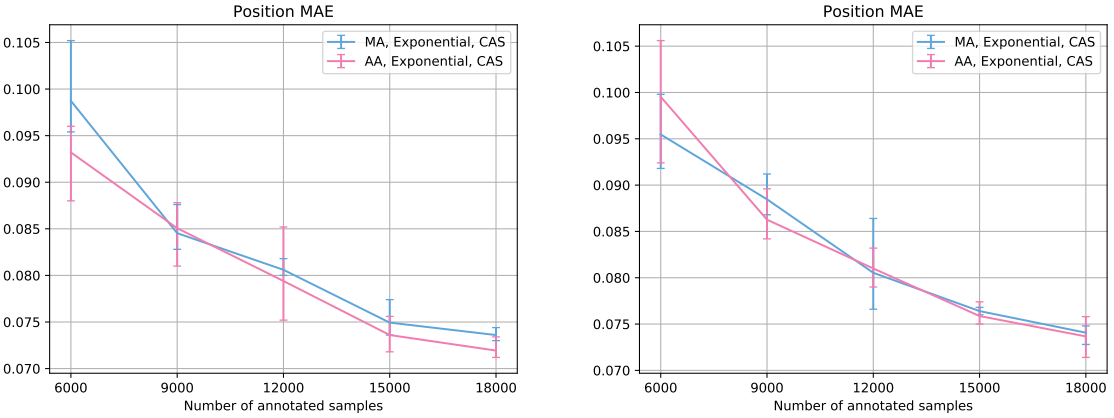
5.1.3 Alternative Approach

Experiments with exponential growth for unmatched detections and CAS were made using AA to compare its performance to MA. As seen in Figure 5.6, the AP of using AA is similar to using MA. The same holds for position MAE, according to Figure 5.7, even though there are spread results. Figure 5.8 shows a slightly better performance on F1 score for vehicles using MA. F1 score for pedestrians (Figure 5.9) shows better performance using MA towards the last AL iterations while the results for bicyclists (Figure 5.10) are rather ambiguous.



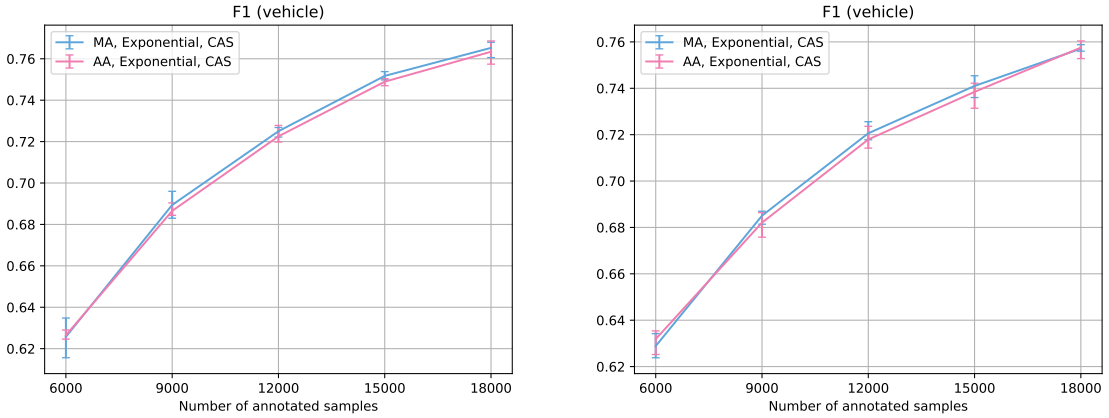
(a) Random seed 1 **(b)** Random seed 2

Figure 5.6: AP. Comparison between MA and AA.



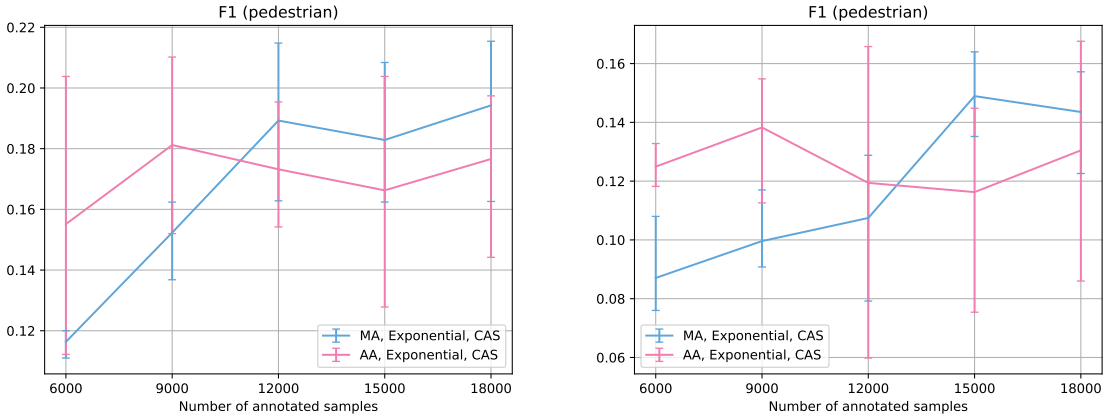
(a) Random seed 1 **(b)** Random seed 2

Figure 5.7: Position MAE. Comparison between MA and AA.



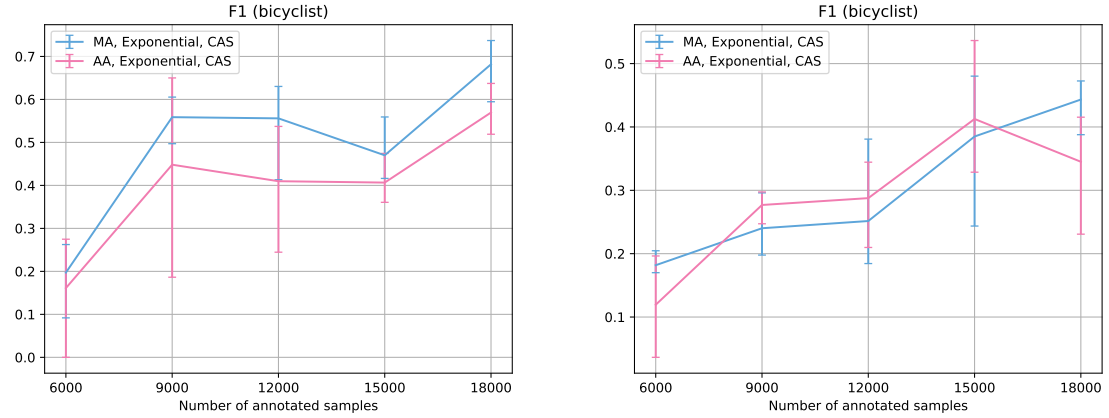
(a) Random seed 1 **(b)** Random seed 2

Figure 5.8: F1 score for vehicles. Comparison between MA and AA.



(a) Random seed 1 **(b)** Random seed 2

Figure 5.9: F1 score for pedestrians. Comparison between MA and AA.



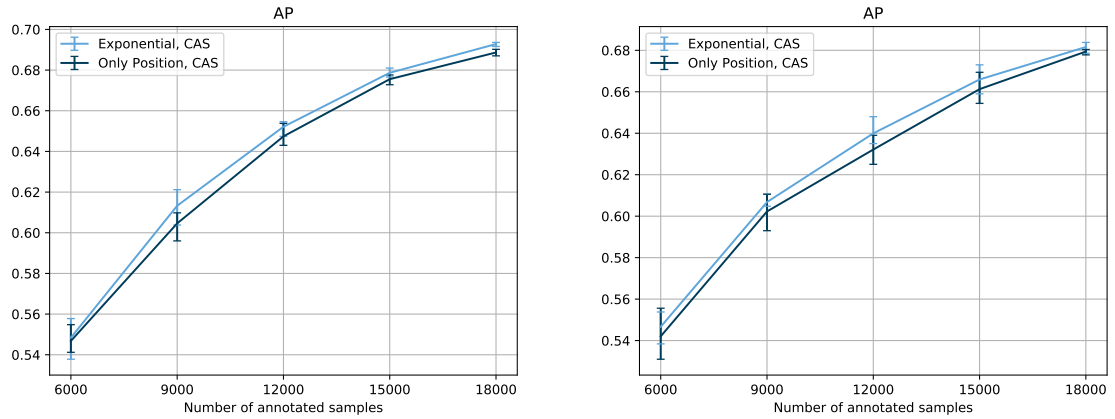
(a) Random seed 1 **(b)** Random seed 2

Figure 5.10: F1 score for bicyclists. Comparison between MA and AA.

5.1.4 Position-only Discrepancy Measure

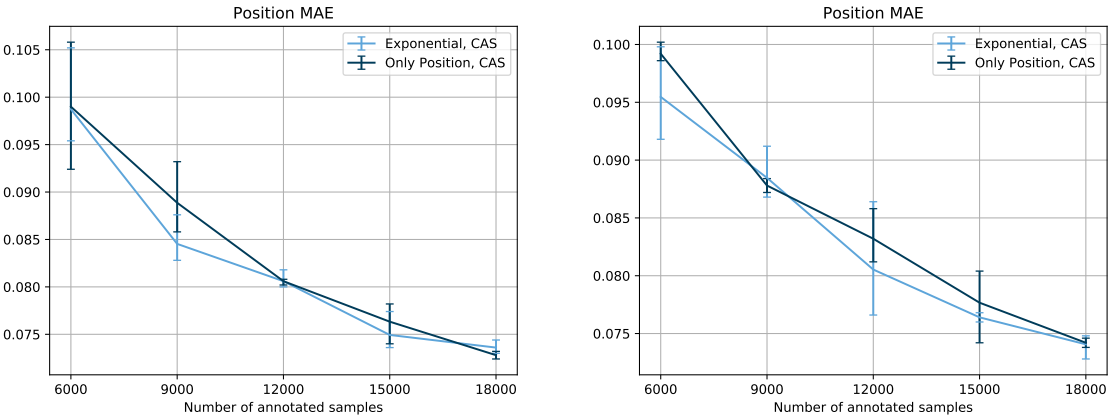
In Section 4.4 we mention that weights can be set to zero to evaluate their importance for the discrepancy measure. One such experiment has been conducted where all weights but w_x, w_y, w_z are set to 0. We do not care about dimension or rotation and we do not measure probability discrepancy nor unmatched detections. The experiment aims to investigate if the performance of position estimation increases if we only consider samples where the position discrepancy is large. It also indicates the influence that the other components such as dimension and unmatched detections have on the performance gain.

Figure 5.11 shows that the AP performance is reduced marginally when using the position-only discrepancy measure, compared to when using the *full* discrepancy measure, i.e. where no weights are set to 0. Figure 5.12 shows similar results for position MAE.



(a) Random seed 1 **(b)** Random seed 2

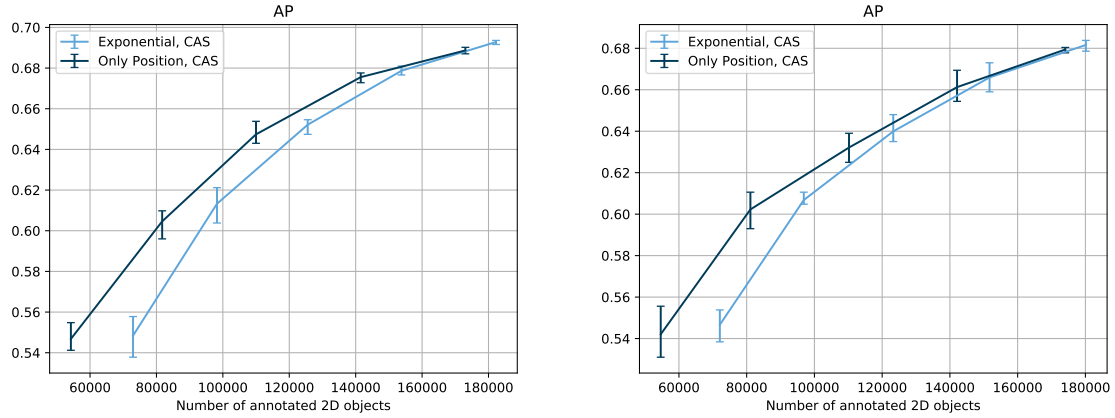
Figure 5.11: AP. Comparison between full discrepancy and position-only discrepancy.



(a) Random seed 1 **(b)** Random seed 2

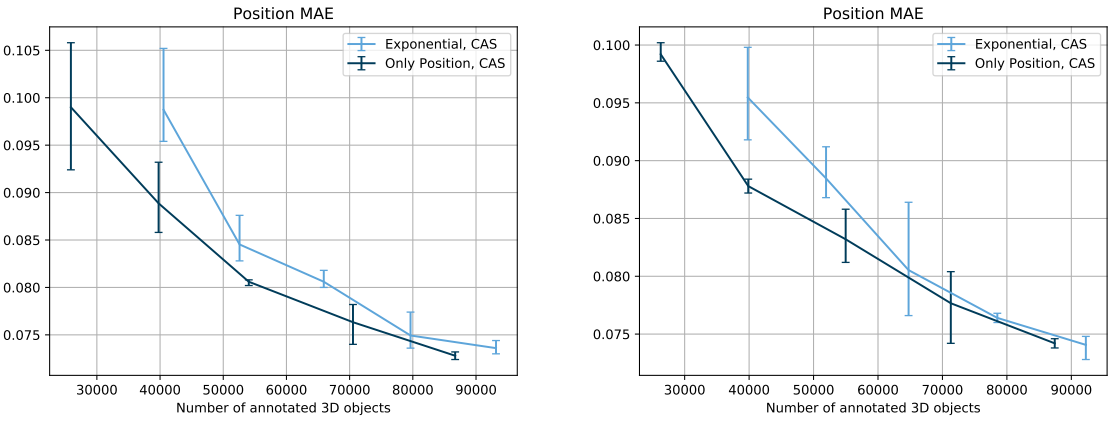
Figure 5.12: Position MAE. Comparison between full discrepancy and position-only discrepancy.

Figure 5.11 and 5.12 show performance with respect to the number of annotated samples. However, if the total annotation cost is to be reduced it is also relevant to look at performance with respect to the number of annotated objects. Annotating a sample that contains many objects takes more time than annotating a sample with few objects. Therefore, Figure 5.13 shows AP per annotated 2D object (includes 3D objects as well) and Figure 5.14 shows position MAE per annotated 3D object. These plots show that the discrepancy measure based solely on position achieves better performance for a fewer number of annotated objects, i.e. most likely to a lower cost, than the full discrepancy measure.



(a) Random seed 1 **(b)** Random seed 2

Figure 5.13: AP per annotated object. Comparison between full discrepancy and position-only discrepancy.



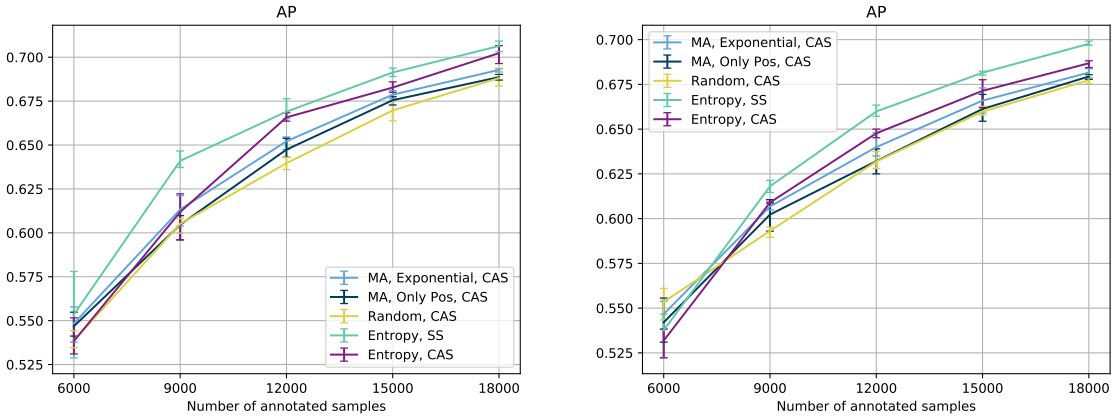
(a) Random seed 1 **(b)** Random seed 2

Figure 5.14: Position MAE per annotated object. Comparison between full discrepancy and position-only discrepancy.

5.1.5 Baseline Comparison

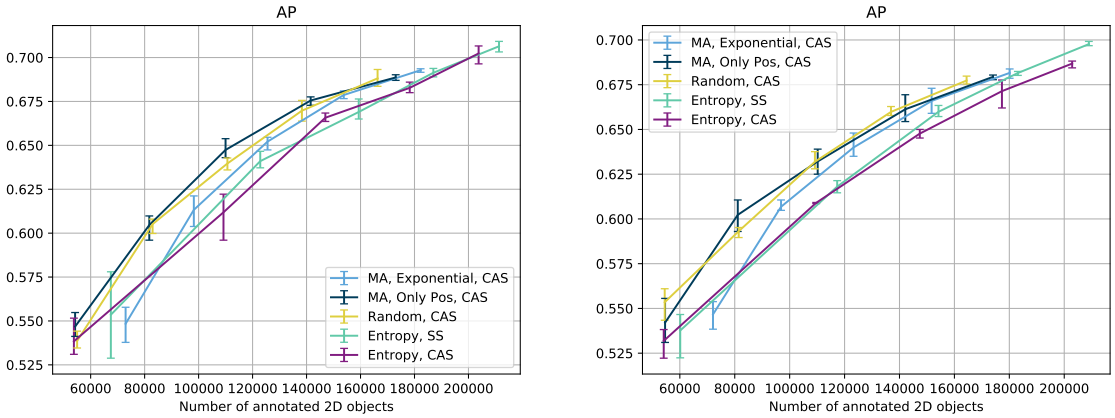
We compare the two sets of parameter settings from Section 5.1.4, i.e. the position-only and the full discrepancy measure, to the entropy and random baseline. We refer to the parameter settings as position-only discrepancy and full discrepancy, respectively.

As shown in Figure 5.15, full discrepancy performs better than the random baseline on AP with respect to the number of annotated samples while position-only discrepancy performs on par with the random baseline. None of the both parameter settings can perform better than the entropy baseline. However, Figure 5.16 shows that position-only discrepancy is superior to full discrepancy on AP with respect to the number of annotated objects. Furthermore, Figure 5.16 also shows that position-only discrepancy performs on par with the random baseline while full discrepancy fails. Both parameter settings perform better than the entropy baseline on AP with respect to the number of annotated objects.



(a) Random seed 1 **(b)** Random seed 2

Figure 5.15: AP. Baselines comparison.

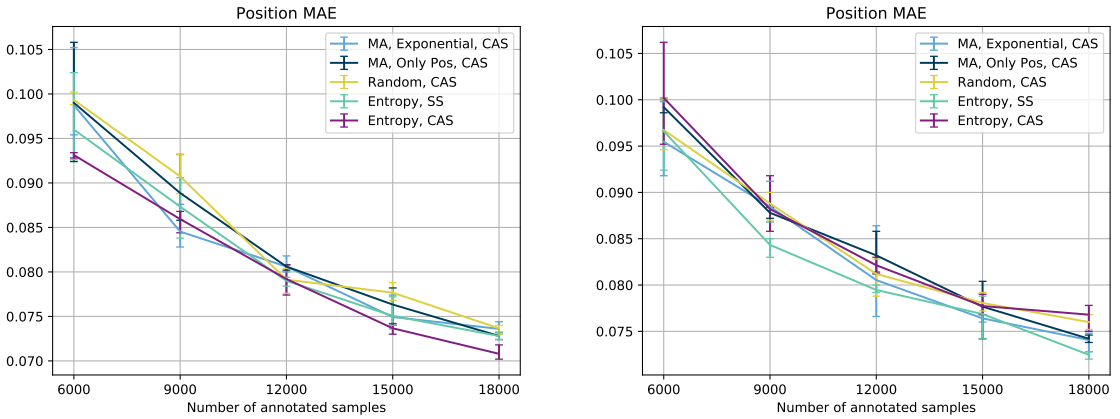


(a) Random seed 1 **(b)** Random seed 2

Figure 5.16: AP per annotated object. Baselines comparison.

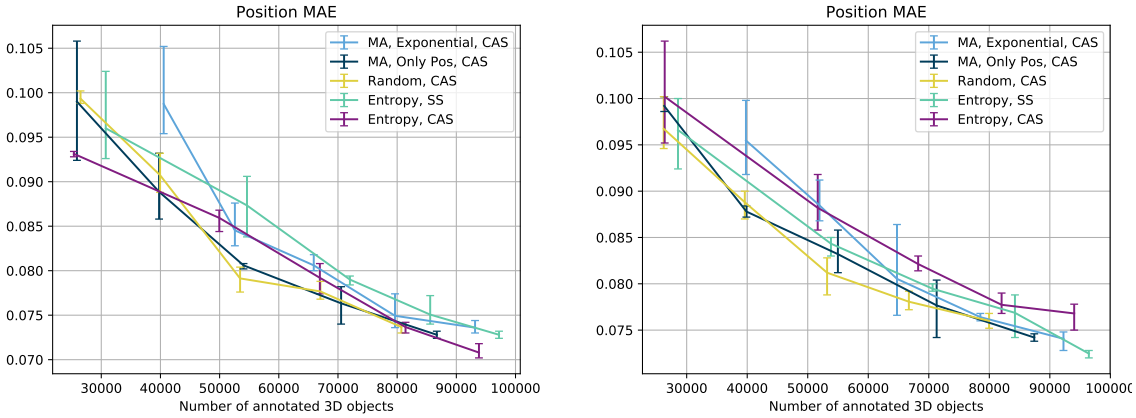
For position MAE with respect to the number of annotated samples, the results

vary a little too much between random seeds to be able to compare the different methods properly, as seen in Figure 5.17. On the other hand, position MAE with respect to the number of annotated objects shows more clear results. Figure 5.18 shows that position-only discrepancy performs better than full discrepancy and on par with or better than the entropy baseline on position MAE with respect to the number of annotated objects. Furthermore, position-only discrepancy performs on par with the random baseline.



(a) Random seed 1 **(b)** Random seed 2

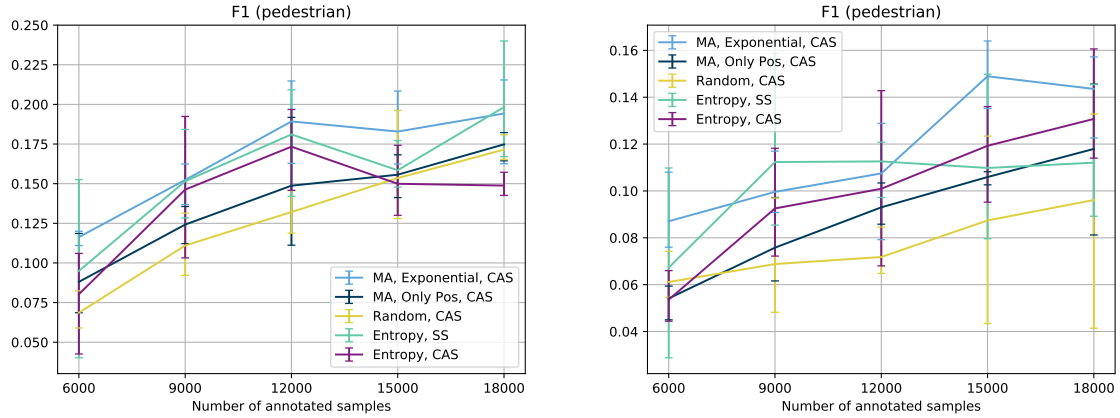
Figure 5.17: Position MAE. Baselines comparison.



(a) Random seed 1 **(b)** Random seed 2

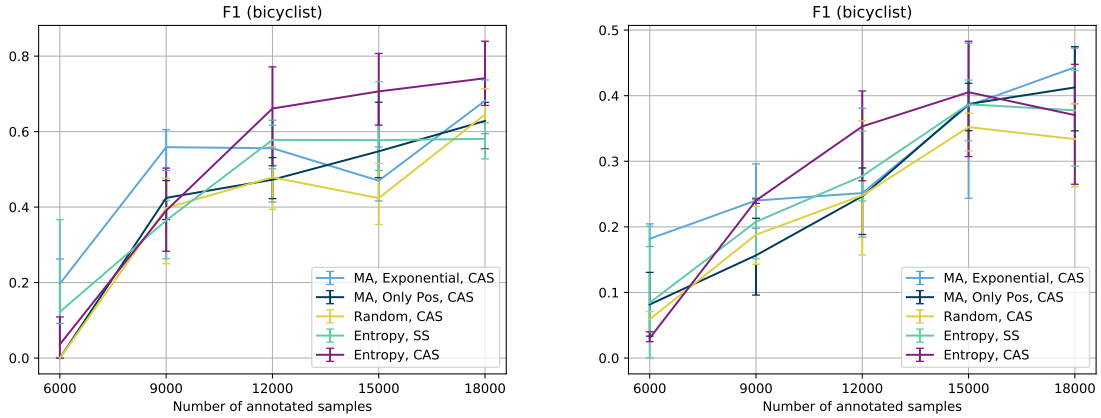
Figure 5.18: Position MAE per annotated 3D object. Baselines comparison.

Figure 5.19 and 5.20 show that full discrepancy outperforms the random baseline on F1 score for pedestrians and bicyclist with respect to the number of annotated samples. Position-only discrepancy performs slightly better than or on par with the random baseline for those KPIs. On the other hand, consistently with previous results, position-only discrepancy perform better than full discrepancy with respect to the number of annotated objects on F1 score for pedestrians and bicyclists, as seen in Figure 5.22 and 5.23. Figure 5.22 and 5.23 also show that position-only discrepancy is able to perform on par with the random baseline in most of the cases.



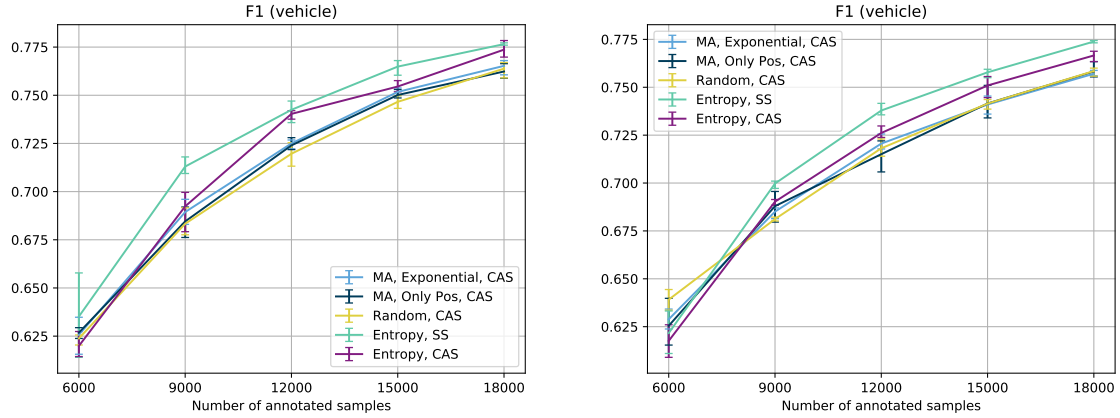
(a) Random seed 1 **(b)** Random seed 2

Figure 5.19: F1 score for pedestrians. Baselines comparison.



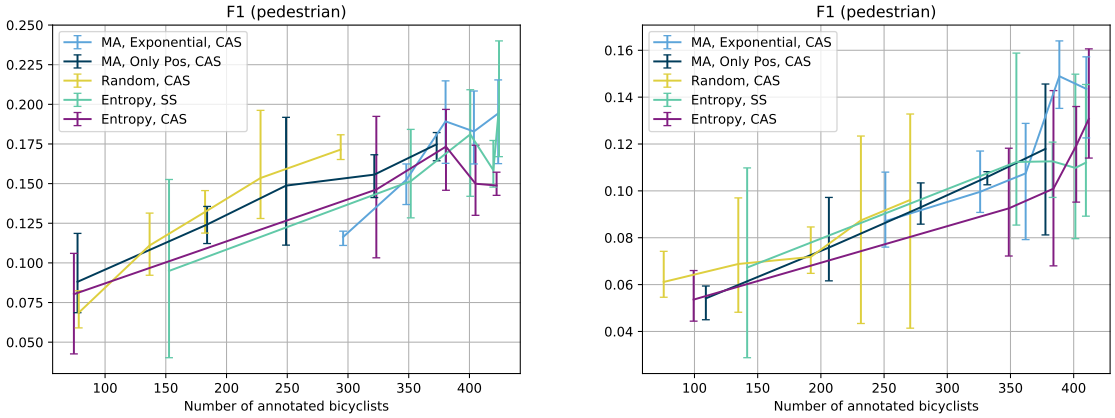
(a) Random seed 1 **(b)** Random seed 2

Figure 5.20: F1 score for bicyclists. Baselines comparison.



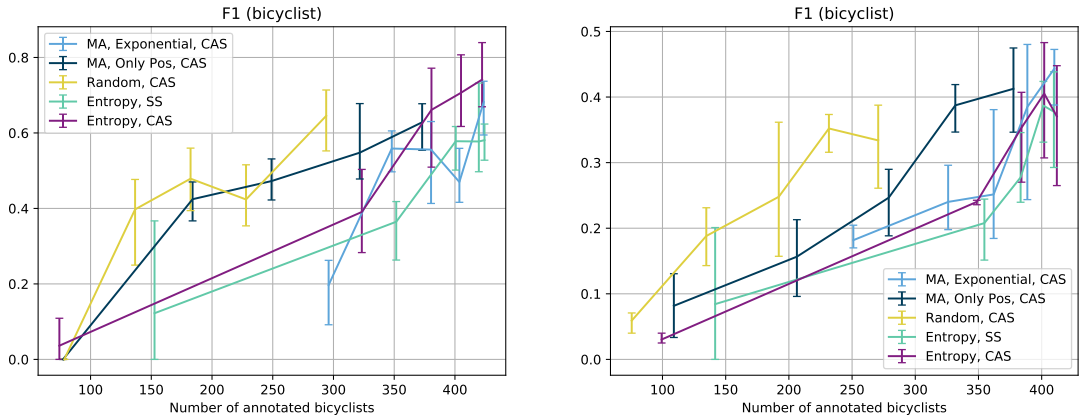
(a) Random seed 1 **(b)** Random seed 2

Figure 5.21: F1 score for vehicles. Baselines comparison.



(a) Random seed 1 **(b)** Random seed 2

Figure 5.22: F1 score for pedestrians, per annotated pedestrian. Baselines comparison.



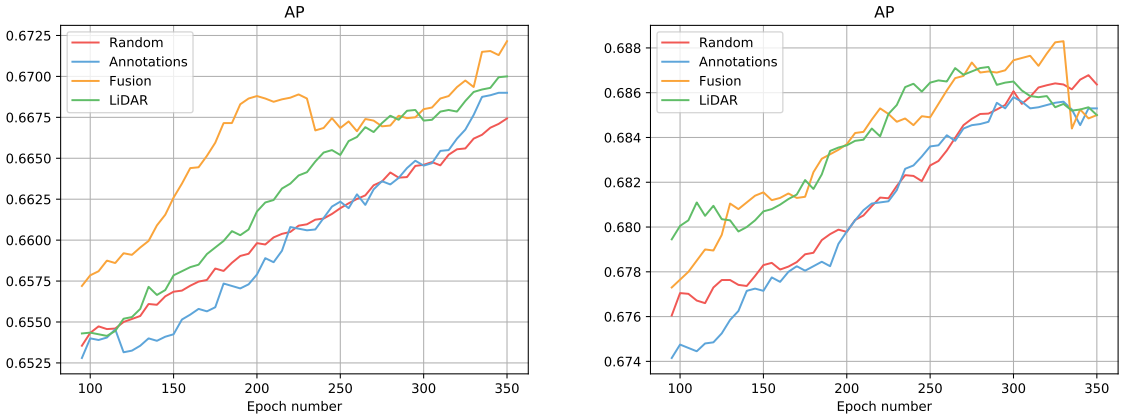
(a) Random seed 1 **(b)** Random seed 2

Figure 5.23: F1 score for bicyclists, per annotated bicyclist. Baselines comparison.

5.2 Fusion Network Discrepancy

Experiments using $\mathcal{M}_F$ as $\mathcal{M}_S$ have been made to test the approach in a real-case scenario. The data selection and AL setting differs from what is described in Section 4.1 and 4.3. The differences originate from $\mathcal{M}_F$ being a demanding model with a complex architecture that takes time to train. $\mathcal{N}$ (the non-annotated set) is set to be equal to $\mathcal{D}_{sub}^{T}$ (our complete training set) and 10000 samples are initially moved from $\mathcal{N}$ to $\mathcal{A}$ for initial training. The validation set $\mathcal{V}$ is set up as described in Section 4.1. Furthermore, only one AL iteration is run instead of 5 and the budget b is set to 5000. The initial training on $\mathcal{A}$ trains for 500 epochs and the AL iteration trains for 350 epochs.

As described in Section 3.3.2, $\mathcal{M}_F$ functions with or without image input. We want to evaluate the gain of using both images and point clouds over solely point clouds as input to $\mathcal{M}_F$. Therefore, we show results for when $\mathcal{M}_F$ is used with fusion (both images and point clouds) and when it only uses LiDAR (point clouds). We compare to the random baseline (CAS) as well as discrepancy to $\mathcal{M}_A$. The experiments with $\mathcal{M}_A$ as well as $\mathcal{M}_F$ use MA with exponential f and CAS. No experiments are conducted using the position-only discrepancy measure due to a hypothesis that the full discrepancy measure would be most beneficial to test in a real-case scenario.



(a) Random seed 1 **(b)** Random seed 2

Figure 5.24: AP per epoch in one AL iteration. Comparison between Camera-LiDAR fusion, only LiDAR and Annotations.

Figure 5.24 shows good performance for $\mathcal{M}_F$ with respect to number of samples used for training. The results differ between random seeds but they all perform better than the random baseline. Discrepancy with $\mathcal{M}_F$ even outperforms discrepancy with $\mathcal{M}_A$ which is noteworthy. However, by measuring AP with respect to the number of annotated objects instead of annotated samples, the result is different. Figure 5.25 shows AP per 100000 annotated objects after the last epoch. We can then see that $\mathcal{M}_F$ is not as good as $\mathcal{M}_A$ if we want good performance for a low number of annotated objects.

Furthermore, Figure 5.24 shows that using both images and point clouds can help in achieving better performance but the difference to only using point clouds

is sometimes not very large. Figure 5.26 shows performance in position MAE. We cannot draw any conclusions based on position MAE since the results differ between random seeds.

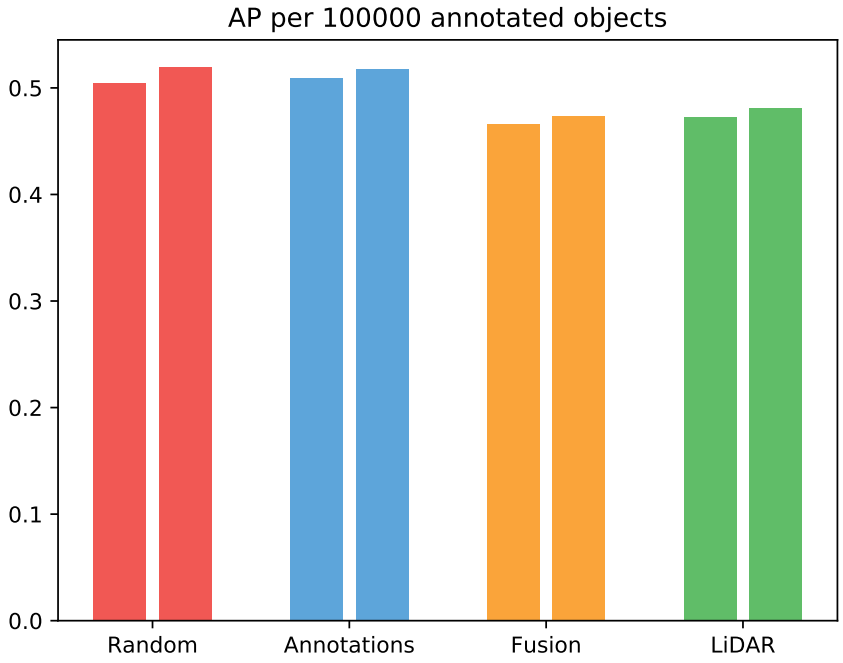
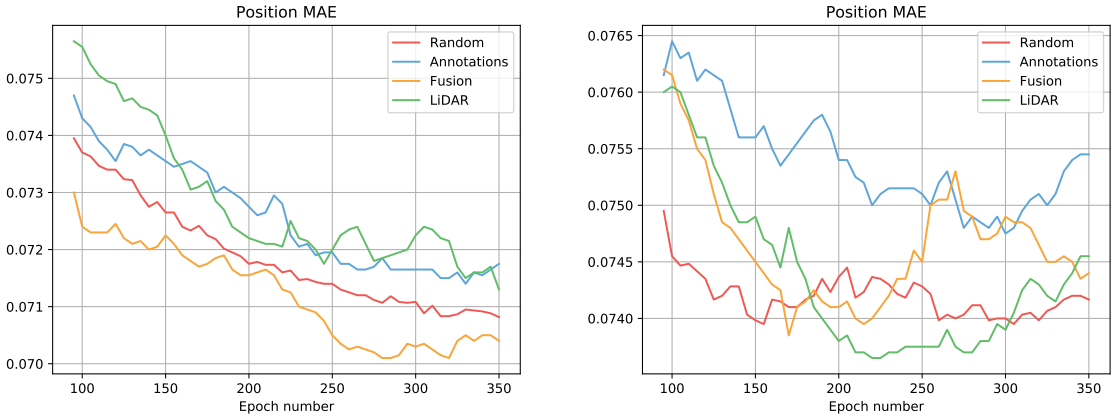


Figure 5.25: AP after last epoch normalized per 100000 annotated objects. Comparison between Camera-LiDAR fusion, only LiDAR and Annotations. Results from two different random seeds are shown.



(a) Random seed 1 **(b)** Random seed 2

Figure 5.26: Position MAE per epoch in one AL iteration. Comparison between Camera-LiDAR fusion, only LiDAR and Annotations.

6
Discussion

Even though the discrepancy method sometimes performs better than the random baseline, the gains are not large. These are rather surprising results since the intuition behind the method is clear. The discrepancy between detections and annotations should intuitively find the samples where the DNN is most inaccurate since we are explicitly comparing to the annotations which the DNN is going to learn from. There are no obvious causes as to why the method is not performing significantly better than random selection.

There has been zero to little success in finding out exactly what is needed for the discrepancy method to perform better. However, in the following sections, ideas on what can be done differently to achieve new results are presented.

We begin by discussing the dataset and the two selection strategies in Section 6.1. The discrepancy method is discussed and questioned in Section 6.2. Section 6.3 discusses the performance of the entropy baseline while Section 6.4 highlights the need of a proper annotation cost model. Lastly, further potential improvements and future research are discussed in Section 6.5.

6.1 Dataset and Selection Strategy

One way of changing the experiment setting is to use a dataset where the training and validation sets are more similar to each other. As described and motivated in Section 4.1, the validation set used in the experiments consists of images taken from one angle while the training set contains images from multiple angles. This difference between training and validation sets is also the original cause to why the CAS strategy is invented. The SS strategy selects significantly fewer images from the front camera than from any other angle. Intuitively, this means that $\mathcal{M}_P$ then performs poorly on the angles that are selected. The reason for the poor performance could be because of the dataset being biased towards some other angle. However, this needs further investigation.

One could argue that the discrepancy method with SS is working just fine since it selects samples where $\mathcal{M}_P$ needs to improve. If tested on a dataset where the training and validation sets are more representative of each other, e.g. contains images from one single angle, the SS strategy might work better.

6.2 The Discrepancy Method

Our result presents two different discrepancy measures with different advantages. The full discrepancy measure has in general good a performance with respect to the number of annotated samples. It performs better than or on par with the random baseline on several KPIs. However, it fails to perform on par with the random baseline with respect to the number of objects.

Furthermore, the position-only discrepancy measure has lower performance than the full discrepancy measure with respect to the number of annotated samples. However, the position-only discrepancy measure is still able to perform on par with the random baseline and does so also with respect to the number of annotated objects. These results make the position-only discrepancy a middle way that can perform on par with the random baseline on most KPIs but never outperforms anything.

6.2.1 Main and Alternative Approach

The comparison between MA and AA in Section 5.1.3 shows that there is no difference KPI performance-wise between both approaches. Due to the hypothesis that the full discrepancy measure is superior to the position-only discrepancy measure, no comparison is done between MA and AA with the usage of the position-only discrepancy measure. However, based on the results shown in Section 5.1.5, we can not dismiss the hypothesis that a comparison between MA and AA using the position-only discrepancy measure would show larger differences between both approaches. Nevertheless, that is left for future work.

6.2.2 Unmatched Detections

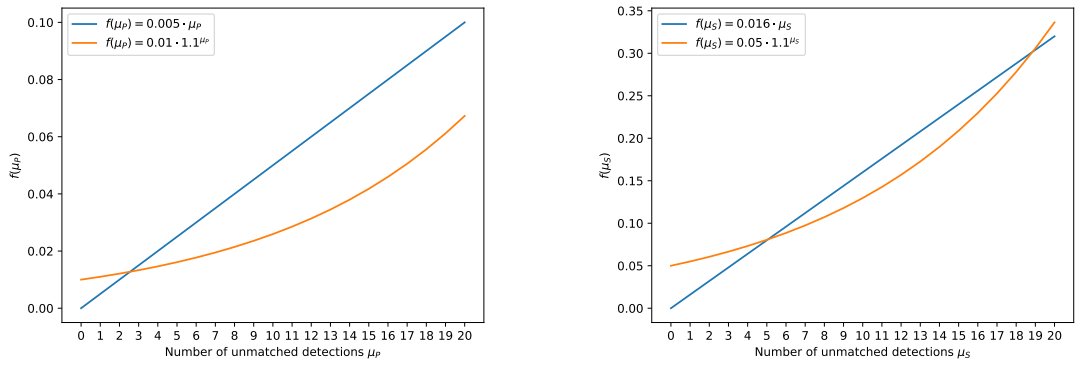
The exact reasons why the full discrepancy measure works better in some cases, and why the position-only discrepancy measure is a mediocre middle way, are hard to know without doing more experiments. Trying out more combinations of different parameter settings could shine more light on the reasons for the difference in performance. However, there is already a plausible cause to the difference.

It is clear, from Figure 5.16, that the full discrepancy measure selects samples containing more objects than the position-only discrepancy measure as well as the random baseline. The reason for this is believed to be the unmatched detections. By looking at examples such as Figure 3.5, we note that samples with many objects also tend to have many unmatched detections. The connection between many objects and many unmatched detections is most likely due to $\mathcal{M}_P$ having a higher precision in environments with fewer objects. Furthermore, the fact that the position-only discrepancy measure selects samples with a lower number of objects indicates that the unmatched detections theory is correct since the position-only discrepancy measure does not count unmatched detections.

Figure 5.24 shows a noteworthy result where the discrepancy method using $\mathcal{M}_F$ outperforms the discrepancy method using $\mathcal{M}_A$. Again, the reason is because of selecting samples with many objects as seen in Figure 5.25. For samples containing

many objects, both $\mathcal{M}_P$ and $\mathcal{M}_F$ make many detections. The high number of detections leads to a higher risk of unmatched detections. Also, from experience with $\mathcal{M}_F$, we know that $\mathcal{M}_F$ sometimes detects more objects than there are annotations in the sample. The obvious path from here is to test the position-only discrepancy measure using $\mathcal{M}_F$. However, as mentioned in Section 5.2, no such experiments are done. All experiments are done with the hypothesis that the full discrepancy measure is the single best approach, which is shown not to be entirely correct. Due to the architecture of $\mathcal{M}_F$ being quite complex (see appendix A.2), experiments are time-consuming. Consequently, new time-consuming experiments with $\mathcal{M}_F$ are not done.

Assuming that the unmatched detections theory is correct, there are several alternatives to be considered. Different functions and weights for the number of unmatched detections can be tested. Section 5.1.2 already presents a comparison between two different functions where the function has very little impact on the performance. The both functions are plotted in Figure 6.1. The weights are chosen subjectively based on example images, as described in Section 3.7. Further experiments can be done to find better weights that do not force the method to select samples with many objects. Furthermore, Figure 6.1 shows that the selected weights make the tested functions quite similar for the first 20 unmatched detections.



(a) Weighted for unmatched detections from $\mathcal{M}_P$, i.e. using w_P.

(b) Weighted for unmatched detections from $\mathcal{M}_S$, i.e. using w_S.

Figure 6.1: Plots of the linear and exponential functions for unmatched detections used in the experiments.

Another way of counting unmatched detections is to make it relative to the total number of detections in the sample. In other words, for sample $\mathbf{X}$, the discrepancy in unmatched detections from the primary and secondary network is calculated

$$\frac{\mu_P}{|\mathcal{X}_P|} \quad \text{and} \quad \frac{\mu_S}{|\mathcal{X}_S|} \tag{6.1}$$

, respectively. Recall that $\mathcal{X}_P$ and $\mathcal{X}_S$ are the sets of detections from the primary and secondary networks, respectively. This would then compensate for the factor that samples containing many objects tend to have many unmatched objects.

Alternatives to counting *all* unmatched detections can also be investigated. One example is to use the objectness probabilities (see Section 3.5.1) of the unmatched

detections. Unmatched detections with high objectness would then contribute more to the discrepancy than unmatched detections with low objectness. By including the objectness, the unmatched detection with high objectness is more likely to be a detection we care about, i.e. either a missed detection or a false positive. Detections with low objectness are most likely not detections of real objects.

Furthermore, the results for the position-only discrepancy measure might argue that the unmatched detections should not be taken into consideration at all. However, that would probably make it harder for the discrepancy method to improve precision and recall, and thereby AP and F1 score.

6.2.3 F1 Score Gain

As shown in Section 5.1.5, compared to the random baseline, the full discrepancy measure significantly improves the F1 score performance on pedestrians and bicyclists with respect to the number of samples. However, as expected, it does so by selecting samples with more pedestrians and bicyclists. We believe that this is due to the discrepancy method selecting samples containing many objects, as explained in Section 6.2.2. Those samples tend to be city images where pedestrians and bicyclists by nature occur more frequently than in highway images which, in general, have fewer objects in them. This hypothesis is supported by the fact that the position-only discrepancy measure does not show as large gains on F1 score for pedestrians and bicyclists.

Furthermore, the fluctuating performance in these results makes it hard to draw any supported conclusions. The fluctuating performance can be the cause of too few pedestrians and bicyclists in the validation set. Therefore, the fluctuation could be prevented by increasing the size of the validation set. However, that would slow down the experiment time remarkably.

6.2.4 Position-only Discrepancy Measure

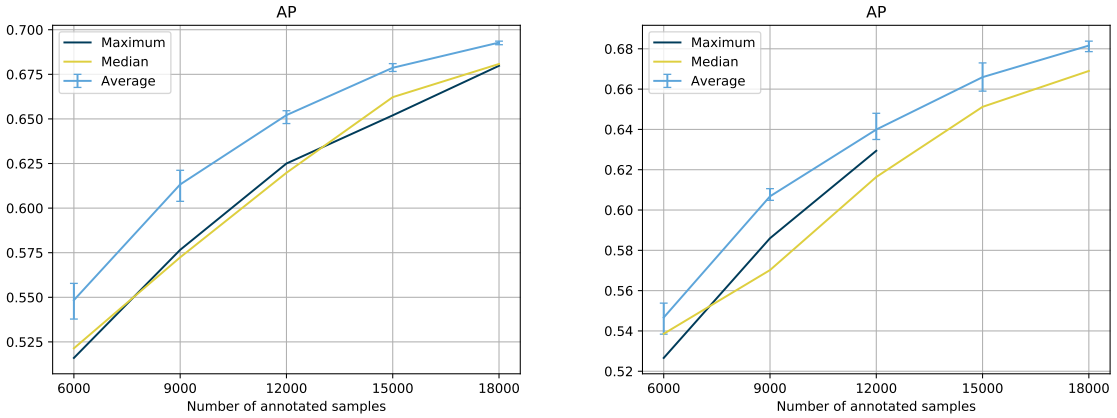
As described in Section 5.1.4, the position-only experiment aims to investigate if the performance of position estimation improves if we only consider samples where the position discrepancy is large. The experiment is conducted due to the intuition that the LiDAR can help increasing the performance in position. However, the result in Figure 5.17 shows that there is no significant improvement.

A further investigation including the other components could be beneficial. For example, only considering position and dimension in the discrepancy to investigate if the performance of the dimension improves.

6.2.5 Discrepancy Aggregation for Matched Detections

As shown in equation 3.21, the discrepancies for all matched detections are aggregated using an average-operation. However, the usage of average for aggregation can be sensitive to outliers having a very large or very small discrepancy. To confirm that average is a good choice, experiments are also made using median and maximum as aggregation functions. The experiments all use full discrepancy settings, i.e. the curve for average is identical to the curve for the full discrepancy in Figure

5.11. The experiments that use median and maximum for aggregation are only run once for each random seed. The results for AP are shown in Figure 6.2. Note that, due to unforeseen reasons, the experiment on the second random seed that uses maximum as aggregation function, unfortunately, stopped before it finished. Still, the plots show that average is the best choice. However, the reason for why the average aggregation performs best needs further investigation.



(a) Random seed 1 **(b)** Random seed 2

Figure 6.2: AP. Comparison between three different aggregation functions for the discrepancy of matched detections.

6.3 Entropy Baseline

As presented in Section 5.1.5, the entropy baseline performs very well with respect to the number of samples but poorly with respect to the number of objects. Meanwhile, equation (4.4) shows that the entropy baseline uses an informativeness score that is directly proportional to the number of objects in the sample. The proportionality explains the selection of samples with many objects and the superior performance compared to the other methods with respect to the number of samples. It further motivates the use of a different entropy baseline, that aggregates the entropy values of each object in a sample with, for example, average, max or median, instead of a sum. Experiments using such an entropy baseline are not conducted and are left for future work.

6.4 Annotation Cost Model

Our results are analyzed with respect to the number of annotated samples as well as the number of annotated objects. We also state that annotating a sample with few objects is cheaper than annotating a sample with many objects. This is most likely true. However, it might not be cheaper to annotate, for example, five samples containing one object each, than annotating one sample containing five objects.

If the objective of a study, like the one presented in this thesis, is to minimize annotation cost while maximizing DNN performance, then an annotation cost model

is necessary. Preferably, the cost model should also be realistic if the study is to be applied to real-case scenarios. We draw most of our conclusions based on a cost model that is linearly dependent on the number of annotated objects. Such a cost model is, as described in the previous paragraph, not always entirely realistic. It is, however, more realistic than a cost model solely based on the number of annotated samples.

We believe that a crucial part in developing this thesis further, with the goal of being useful in real-case scenarios, is to have a better annotation cost model. Such a cost model is suggested to be based on studies investigating how the number of objects and samples are correlated to annotation cost.

6.5 Future Research

Due to the minimal gain in performance, the way we calculate discrepancy might be questioned. As described in Section 3.7, the process of choosing the components and parameter settings of the discrepancy measure is heavily built on educated guesses on what $\mathcal{M}_P$ will benefit from learning. Furthermore, calculating the discrepancy to cover the inaccuracy in all attributes in only one combined value could be a weak informativeness score.

Instead, another approach where we select the largest discrepancy for each component could be feasible. We have shown that there is a potential gain in selecting samples that only has a large discrepancy in position. Therefore, we could select a fixed proportion of samples with a large discrepancy in position, a fixed proportion of samples with a large discrepancy in dimension, and so forth. In that way, we could find particular samples that lack performance for each component.

Moreover, a deeper investigation of what type of samples are selected by the discrepancy method should be done. For example, investigate how diverse the set of selected samples is, e.g. if the set only contains mostly scenes from highways or cities. This could be done using clustering methods and compare to the cluster of randomly selected samples.

Another way of utilizing the LiDAR for AL is to measure some kind of Expected Model Change (see Section 2.2). The detections from $\mathcal{M}_S$ will then serve as annotations and the gradient of the loss compared to detections from $\mathcal{M}_P$ and $\mathcal{M}_S$ will be measured. One benefit compared to the discrepancy method is that we do not need to arbitrarily adjust weights. To start with, the method could be tested by comparing it to annotations, as done in this thesis.

Furthermore, detections from $\mathcal{M}_S$ can also be used for semi-supervised learning. That is, learn directly from the detections from $\mathcal{M}_S$. In particular, since $\mathcal{M}_S$ is assumed to be accurate in position, we could specifically try to improve the position performance. More specifically, for the non-annotated samples which are annotated using detections from $\mathcal{M}_S$, we set the loss to 0 on all other components except position. In that way, we could improve the position accuracy for $\mathcal{M}_P$ with samples that are not annotated.

7
Conclusion

This thesis presents an Active Learning algorithm for 3D Object Detection that compares detections from two Deep Neural Networks. One network accepts images as input, while the other one accepts point clouds from a LiDAR sensor. The Active Learning algorithm selects samples where the detections from both networks differ the most according to a discrepancy measure. The discrepancy method for Active Learning appears to be a promising method but it requires more research to be of use in real-case scenarios.

The discrepancy measure matches detections from both networks with each other to quantify their discrepancy. Some detections can be left unmatched and can be included in the discrepancy measure by measuring the number of unmatched detections. We can show that a discrepancy measure can perform better than selecting samples at random when measuring the number of unmatched detections, in addition to the discrepancy in position, dimension, rotation, and class prediction. However, this method fails to maintain the annotation cost compared to selecting samples at random, since the method selects samples containing a larger number of objects. Since samples with more objects are expected to take more time to annotate, this corresponds to an increased annotation cost per image. By contrast, a discrepancy measure that is solely position-based performs on par with random selection and maintains the annotation cost.

To make the discrepancy method useful in real-case scenarios, a better way of counting unmatched detections is believed to be required. Furthermore, the method needs to be tested on an additional dataset to properly verify its potential.

Bibliography

[1] A. Rosebrock, "A visual equation for Intersection over Union (Jaccard Index)," http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, 2016, licensed under the Creative Commons Attribution-Share Alike 4.0 International license, [Online; accessed 23-April-2020].

[2] Walber, "Precision and recall," https://commons.wikimedia.org/wiki/File:Precisionrecall.svg, 2014, licensed under the Creative Commons Attribution-Share Alike 4.0 International license, [Online; accessed 29-April-2020].

[3] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[4] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by Committee," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: Association for Computing Machinery, 1992, p. 287–294. [Online]. Available: https://doi.org/10.1145/130385.130417

[5] Y. Siddiqui, J. Valentin, and M. Nießner, "ViewAL: Active Learning with Viewpoint Entropy for Semantic Segmentation," 2019.

[6] C.-C. Kao, T.-Y. Lee, P. Sen, and M.-Y. Liu, "Localization-Aware Active Learning for Object Detection," in *Computer Vision – ACCV 2018*, C. Jawahar, H. Li, G. Mori, and K. Schindler, Eds. Cham: Springer International Publishing, 2019, pp. 506–522.

[7] D. Feng, X. Wei, L. Rosenbaum, A. Maki, and K. Dietmayer, "Deep Active Learning for Efficient Training of a LiDAR 3D Object Detector," *CoRR*, vol. abs/1901.10609, 2019. [Online]. Available: http://arxiv.org/abs/1901.10609

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

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

[10] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[11] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-View 3D Object Detection Network for Autonomous Driving," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[12] L. Ramshaw and R. E. Tarjan, "A Weight-Scaling Algorithm for Min-Cost Imperfect Matchings in Bipartite Graphs," in *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, 2012, pp. 581–590.

[13] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109

[14] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. ACM*, vol. 19, no. 2, p. 248–264, Apr. 1972. [Online]. Available: https://doi.org/10.1145/321694.321699

[15] W. Zumino, "Intuition behind the Hungarian Algorithm," https://brilliant.org/discussions/thread/intuition-behind-the-hungarian-algorithm/, 2017, [Online; accessed 23-May-2020].

[16] K. G. Murty, *Network Programming*. USA: Prentice-Hall, Inc., 1992. [Online]. Available: http://www-personal.umich.edu/~murty/books/network_programming/

[17] J. Hui, "mAP (mean Average Precision) for Object Detection," https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, 2018, [Online; accessed 24-April-2020].

[18] N. Chinchor, "MUC-4 Evaluation Metrics," in *Proceedings of the 4th Conference on Message Understanding*, ser. MUC4 '92. USA: Association for Computational Linguistics, 1992, p. 22–29. [Online]. Available: https://doi.org/10.3115/1072064.1072067

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[20] J. Berntsson and A. Tonderski, "Sensor Fusion with Failure Robustness," 2019, submitted.

[21] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[22] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep Continuous Fusion for Multi-Sensor 3D Object Detection," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[23] P. J. Huber, *Robust Estimation of a Location Parameter*. New York, NY: Springer New York, 1992, pp. 492–518. [Online]. Available: https://doi.org/10.1007/978-1-4612-4380-9_35

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[25] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297

A

Appendix 1

A.1 Matching Example

The following matrices correspond to the problem instance shown in Figure 3.3.

$$M_{IoU} = \begin{pmatrix} 0 & 0.09 & 0 & 0 & 0 & 0 & 0.65 \\ 0.88 & 0 & 0.04 & 0 & 0 & 0 & 0 \\ 0.01 & 0 & 0 & 0.76 & 0 & 0 & 0 \\ 0 & 0.61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.69 & 0 & 0 \\ 0.04 & 0 & 0.74 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.66 & 0 \end{pmatrix} \tag{A.1}$$

$$M_{obj} = \begin{pmatrix} 0.99 & 0.98 & 0.97 & 0.96 & 0.94 & 0.94 & 0.93 \\ 0.99 & 0.98 & 0.97 & 0.96 & 0.94 & 0.94 & 0.93 \\ 0.99 & 0.98 & 0.97 & 0.96 & 0.94 & 0.94 & 0.93 \\ 0.99 & 0.98 & 0.97 & 0.96 & 0.94 & 0.94 & 0.93 \\ 0.99 & 0.98 & 0.97 & 0.96 & 0.94 & 0.94 & 0.93 \\ 0.99 & 0.98 & 0.97 & 0.96 & 0.94 & 0.94 & 0.93 \\ 0.99 & 0.98 & 0.97 & 0.96 & 0.94 & 0.94 & 0.93 \end{pmatrix} \tag{A.2}$$

$$M_{depth} = \begin{pmatrix} 0.27 & 0.73 & 0.80 & 0.57 & 0.73 & 0.61 & 0.99 \\ 0.94 & 0.40 & 0.23 & 0.51 & 0.21 & 0.48 & 0.29 \\ 0.53 & 0.70 & 0.41 & 0.91 & 0.37 & 0.84 & 0.51 \\ 0.33 & 0.88 & 0.66 & 0.68 & 0.60 & 0.73 & 0.82 \\ 0.17 & 0.46 & 0.79 & 0.36 & 0.87 & 0.39 & 0.64 \\ 0.25 & 0.66 & 0.89 & 0.51 & 0.81 & 0.55 & 0.91 \\ 0.45 & 0.83 & 0.49 & 0.93 & 0.44 & 1.00 & 0.60 \end{pmatrix} \tag{A.3}$$

$$\begin{aligned} M &= M_{IoU} \odot M_{obj} \odot M_{depth} \\ &= \begin{pmatrix} 0 & 0.06 & 0 & 0 & 0 & 0 & 0.60 \\ 0.83 & 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0.01 & 0 & 0 & 0.66 & 0 & 0 & 0 \\ 0 & 0.52 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.57 & 0 & 0 \\ 0.01 & 0 & 0.64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.62 & 0 \end{pmatrix} \end{aligned} \tag{A.4}$$

After filtering with $\tau_{IoU} = 0.2$ and $\tau_{depth} = 0.6$, we obtain

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.60 \\ 0.83 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.66 & 0 & 0 & 0 \\ 0 & 0.52 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.57 & 0 & 0 \\ 0 & 0 & 0.64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.62 & 0 \end{pmatrix} \tag{A.5}$$

which gives us the matchings shown in Figure 3.3 after using Hungarian method. Note that many zeros are introduced which makes (A.5) an easy match task. The match task could be much trickier if there are several bounding box overlaps when calculating IoU, as in Figure 3.4.

A.2 Architecture of Camera-LiDAR Fusion Network

The architecture of the fusion DNN is described as follows. First, depth for every pixel in the image, referred to as depth map, is estimated using ResNet-18 [9]. An example of an estimated depth map with its target can be seen in Figure A.1. The estimated depth map is transformed into an estimated point cloud, by unprojecting each pixel. The point cloud created by the estimated depth map is called *pseudo point cloud*. The pseudo point cloud is created by first increasing the resolution of the original input size using a Feature Pyramid [21]. In this way, the number of points in the pseudo point cloud can be adjusted to a similar number of points in the point cloud. Each point in the pseudo point cloud as 3D coordinates $(x, y, z)^T$ is concatenated with high-level image features f as $p = (x, y, z, f)^T$. The high-level image features are obtained from the last layers of ResNet-18 when estimating the depth map.

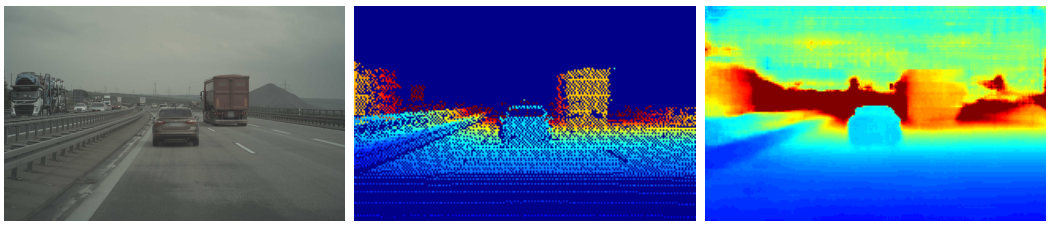


Figure A.1: Example of depth map. Left: Input image. Center: Target as projected LiDAR points in image. Right: Estimated depth map. Blue indicates points close to the ego vehicle and red indicates points further away.

The pseudo point cloud is fused with the LiDAR point cloud into a discrete Bird's Eye View (BEV) feature map using a point-wise discretization method. We refer to [22] for details. Several smaller feature maps are also created from the pseudo point cloud using the same method. All the BEV feature maps are used to output 3D detections by a ResNet-based DNN. An example of output detections can be

seen in Figure A.2. Note that the pseudo point cloud in the camera viewing ray is visualized on top of the LiDAR point cloud.

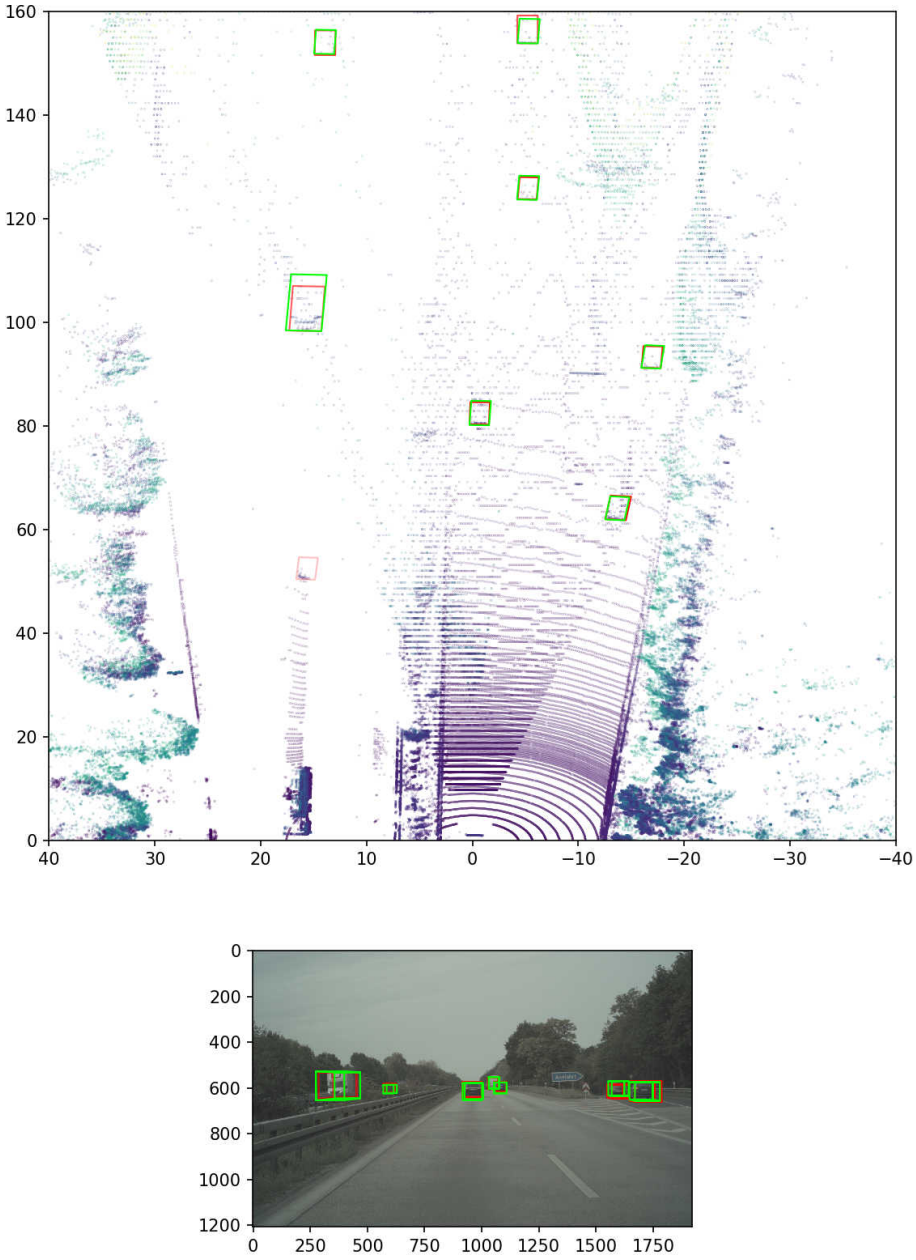


Figure A.2: Example of outputted $\mathcal{M}_F$ detections. Top: BEV with LiDAR and pseudo point cloud. Bottom: Camera view. Red boxes are detections by $\mathcal{M}_F$ and green boxes are the annotated objects.

To detect objects with bounding boxes, regression targets are needed. A regression target is a bounding box expressed as location, dimension, and rotation which the DNN uses as target when learning. Each regression target is placed in a BEV map containing BEV cells. Having $l = (x, y, z)^T$, $d = (w, h, l)^T$ and r from annotations, the regression targets are encoded by

$$l_e = l - l_a \tag{A.6}$$

$$d_e = \log \frac{d}{d_a} \tag{A.7}$$

$$r_e = (\cos 2r, \sin 2r) \tag{A.8}$$

where l_a is the center location of each BEV cell and d_a is the average dimension in the whole dataset. By using sine and cosine in the encoding for rotation, the periodic value jump is counteracted. To make the encoding invariant to heading direction of the object, the rotation is multiplied with a factor two. As a final parameter, a confidence score is predicted in order to determine if there actually is an object in that bounding box. If there exist an object in the bounding box, the target for the confidence score is 1.

A weighted combination of object detection losses gives the total loss

$$L = \phi L_{depth} + \gamma L_{reg} + L_{cls} \tag{A.9}$$

where L_{depth} is the loss between the estimated and target depth seen as an example in Figure A.1 and L_{reg} is Huber loss with $\delta = 1$ for regression. For the classification loss, L_{cls} is Focal Cross-entropy loss. Focal loss puts more focus on misclassified objects by reducing the relative loss for well-classified objects. We refer to [23] regarding details about Huber loss and [24] regarding details about Focal Cross-entropy loss.

A.3 Modifications in Camera-LiDAR Fusion Network

The fusion DNN $\mathcal{M}_F$ is originally developed to train on KITTI data [25]. To make it compatible with Zenuity data, a few modifications have to be made. Firstly, the two datasets come in different formats, such as different image resolutions and point clouds in different coordinate systems. Different formats lead to modifications in both pre-processing and post-processing.

Secondly, $\mathcal{M}_F$ is only compatible with images taken by a front camera. The reason is that $\mathcal{M}_F$ filters out all LiDAR points that are not in front of the ego vehicle. To include as many samples as possible for our AL experiments, $\mathcal{M}_F$ has to be compatible with every camera angle. This is done by transforming every LiDAR point from the LiDAR coordinate system into the camera coordinate system. Instead of filtering out all LiDAR points that are not in front of the ego vehicle, we filter out all points that are not in the viewing ray of the camera.

The last step is to make $\mathcal{M}_F$ compatible for the ten different classes in Zenuity data as it is only compatible with binary classification, i.e. background and car. This includes changing Sigmoid to Softmax in the Cross-entropy loss. More importantly, since $\mathcal{M}_F$ uses average dimensions in the regression targets encoder, it is preferable for better performance to have one average dimension for every class, otherwise predicted bounding boxes tend to be the same size regardless class. For example,

a bounding box for a truck should be larger than a bounding box for a car or a pedestrian. We calculate the average dimensions from all the annotated objects in the whole dataset and use these average dimensions when encoding the regression targets.