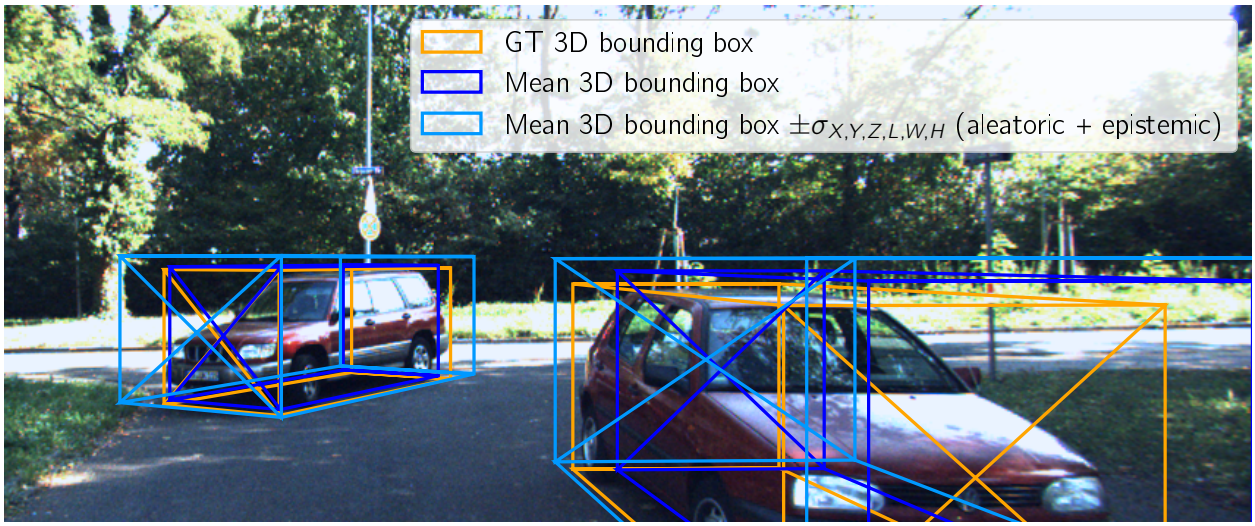
CHALMERS
UNIVERSITY OF TECHNOLOGY



Deep Learning Uncertainties for Monocular 3D Object Detection

Master's thesis in Systems, Control and Mechatronics

Oscar Almér, Emil Andersson

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Deep Learning Uncertainties for Monocular 3D Object Detection

Oscar Almér Emil Andersson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Master's thesis in Systems, Control and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2019

Deep Learning Uncertainties for Monocular 3D Object Detection
Oscar Almér, Emil Andersson

Supervisors: Christoffer Petersson & Hampus Linander, Zenuity AB
Examiner: Lennart Svensson, Department of Electrical Engineering

Master's Thesis 2019
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Signal Processing Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Detected objects and corresponding perception uncertainties visualized in the image plane.

Typeset in LaTeX
Gothenburg, Sweden 2019

Deep Learning Uncertainties for Monocular 3D Object Detection
Oscar Almér, Emil Andersson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In the development of self-driving cars, safety plays a key role. Different sensors are used to continuously interpret the surroundings. Cameras are extremely important and unique in being able to determine context whilst also distinguishing objects with high resolution. However, many of today's machine learning algorithms for processing camera images are unaware of the uncertainties in their predictions and can therefore not be fused optimally with data from other sensors. If these algorithms were to be extended into knowing their uncertainties, they would be of more practical use in safety-critical applications.

In this thesis, we study two methods for estimating uncertainties in machine learning and apply them to a machine learning algorithm for monocular 3D object detection to enable per prediction uncertainty estimates. The methods use ideas from Bayesian statistics to estimate the uncertainties. Furthermore, a framework for propagating uncertainties to the global coordinate system is presented. The first method aims at capturing uncertainties due to noise in the data whilst the other method aims at capturing uncertainties due to imperfections in the model.

The results are evaluated and analyzed extensively with a focus on the usefulness and interpretability of the uncertainty estimates. The presented methods are, to the best of our knowledge, the first of their kind for monocular vision 3D object detection and there is currently no standardized way of evaluating uncertainty estimates. Therefore no comprehensive comparison to other methods could be done. We propose and study ways of evaluating uncertainty estimates. We find that a machine learning algorithm for monocular 3D Object Detection can successfully estimate the uncertainties in its predictions, thus making the algorithm more suitable for usage in a safety-critical application.

Keywords: Machine Learning, Uncertainties, Bayesian, Autonomous Driving, Monocular 3D Object Detection, Aleatoric, Epistemic.

Acknowledgements

We would like to thank Christoffer Petersson and Hampus Linander at Zenuity for giving us the opportunity to perform our master's thesis under their supervision. Thank you for your big engagement in the work and willingness to help. Thank you for all extensive discussions and for truly taking us under your wings.

Moreover, we would also like to thank Lennart Svensson for supervising and examinating our master's thesis constituting our last assignment at Chalmers. Thank you for all the correspondence, extremely valuable feedback and great discussions.

Furthermore, we would like to thank Viktor Skantze, Petter Hansegård, Zeeshan Dar and Hai Dinh constituting our fellow thesis students at the Zenuity Deep Learning division. Thank you for the fantastic company and warming fellowship. You are wonderful people and we are thankful for you being the persons we got to share this experience with.

Finally, we would like to thank everyone at Zenuity for their inexhaustible helpfulness and our friends, families and girlfriends for your patience regarding our absence and sparse notifications this past six months.

Oscar Almér, Emil Andersson, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1
Introduction

In recent years, the interest in self-driving cars has increased drastically, gaining a lot of attention from the media. As the interest has increased so has the amount of resources that is being invested in the development of these cars. With self-driving cars in the transportation system, the infrastructure would have the possibility to be used much more efficiently and number of accidents to be much fewer. The comfort of car users would also increase and time reserved for driving can be spent on other activities.

Today's Autonomous Driving (AD) systems are currently at a stage where the system assists the driver in different ways, e.g. in highway pilot assist, but where the driver remains responsible for monitoring the driving environment and is ultimately responsible for the driving. The next level of AD is a stage where a vehicle drives completely on its own with no driver supervision and primarily where the vehicle takes full responsibility for the driving. There are no such solutions available on the market today. However, most vehicle manufacturers are aiming for this level of AD. For a self-driving car to be able to take full responsibility of the driving and monitoring of the driving environment, as well as to ever have the possibility of being allowed on the roads and gain the trust of the users, the safety needs to be assured.

To develop vehicles that can control themselves in urban traffic is a very challenging task. The vehicle has to constantly scan and interpret its surroundings using several different sensors to be able to make decisions regarding how to control the vehicle in a safe manner. One of the key sensors in interpreting the surroundings are the cameras which are mainly used to identify where the road is and to detect where and what the other road users are. A common method for road and road user identification from camera images is to use machine learning.

1.1 Machine Learning

Machine learning is a popular technology for solving complex tasks by making a computer learn from data instead of explicitly programming a solution. The most commonly used form of machine learning is supervised learning. In Supervised learning a machine learning model is presented with large amounts of data (inputs) and corresponding correct answers (desired outputs) from which the supervised learning model attempts to learn the pattern. When the model has learned a pattern the hope is that the model is able to predict the correct answer to new, unseen, data where the correct answer is desirable, but unknown. More formally, the supervised

learning model attempts at finding the function $f : X \rightarrow Y$ mapping the input data, X, to output data, Y, and where the training set (X, Y) contains N number of training samples $\{(x_1, y_1) \dots (x_N, y_N)\}$.

There are plenty of machine learning models for detecting where the road is and for detecting other road users. The problem of detecting other road users is attacked in the task of 3D object detection.

1.2 3D Object Detection

To enable safe navigation with an autonomous vehicle, its awareness of surrounding road users has to be ensured. Detection of other road users in 3D infers that the estimates regarding a road users whereabouts should be given in world coordinates, i.e. in the same coordinate frame that the ego vehicle moves in. More specifically, the task is to find a box that tightly encloses the road user to, ultimately, be able to determine where it is and where it is not safe to drive. Furthermore, each road user should be classified as e.g. a car or a pedestrian. It is of interest to know what type of object that has been located since a car and a pedestrian have very different characteristics, e.g. a pedestrian might abru ptly change the direction of which it is heading whereas a car has a limited turn rate. Discerning the road users whereabouts can be done more accurately by having a good description of the uncertainties.

1.3 Uncertainties

Uncertainties are used to describe the precision of various things, but for the subject of Autonomous Drive, uncertainties are used to describe the precision of different sensors, measurements and models. The uncertainties give an indication of how much a measurement from a sensor should be trusted. To have a proper uncertainty description accompanied with each measurement is crucial in order to use each sensor to its fullest potential and to make accurate predictions. Each sensor has its own set of strengths and weaknesses and thus, by having a proper uncertainty description for each measurement, we can exploit the strengths of multiple different sensors to make more accurate predictions than we ever could with only one type of sensor.

To make predictions about other road users whereabouts from camera images machine learning models can be used, as discussed in Section 1.2. However, most of these models are unaware of the uncertainties in their predictions. In a regression problem, the output from the network is often presented as a single value without any uncertainties associated to the estimate. In a classification scenario, the output is most often a percentage of how certain the network is that the detected object is of a certain class. However, this percentage is not necessarily based on actual certainty. To maximize the usefulness of the camera as a sensor, the associated uncertainties has to be estimated in a robust way.

There are approaches for extracting uncertainty estimates in machine learning. These approaches have already been applied to the tasks of semantic segmentation, depth estimation, 2D detection and 3D detection using LiDAR information. This

thesis has, however, focused on extracting uncertainties for the more comprehensive task of monocular 3D object detection.

Lastly, having uncertainty estimates accompanied with the output from machine learning algorithms used in 3D Object Detection would open up several use cases for the output. It would enable the information to be fused with information from other sensors, to make for better estimations. The uncertainty estimates can, furthermore, also be utilized offline to find scenarios where the network indicates high uncertainty. The difficulties in these scenarios can be analyzed, identified and learned from by extending the training data set with similar images. This offline usage of the uncertainties to extend the training data set might thus still have a tremendous impact on the online performance.

1.4 Thesis Objectives

There are plenty of benchmarks for which one can compare the performance of a machine learning model to other models. However, these benchmarks focus solely on the pure performance of the models and does not take uncertainty into account. In 3D object detection pure performance is measured by comparing how well the networks best guesses of where the other road users are coincides with the actual whereabouts of the road users, without taking the uncertainties into account.

A machine learning model which is unaware of its uncertainties is of limited use in safety critical applications. Unfortunately, there is currently no standardized way of evaluating the quality of uncertainty estimates in machine learning and thus no way to compare two models to each other. The goal of the thesis is to first find a way to estimate the deep learning uncertainties for monocular 3D object detection (3DOD) to maximize the usefulness of the neural network outputs and to then propose a method for evaluating the uncertainty estimates to be able to compare models to each other.

When considering methods for extracting uncertainties computational efficiency is not considered, i.e. the solution will be suitable for offline usage and not necessarily for implementation in real-time safety critical systems. This thesis is furthermore limited to estimating uncertainties for one type of task, namely 3DOD. This task includes regression problems, find e.g. the height, width and length of a 3D bounding box, as well as a classification problem where each object has to be classified into e.g. pedestrian or car.

1.5 Contributions

This thesis work comprise the following major contributions to the field:

- **Uncertainty estimation for monocular 3D Object Detection** - Uncertainty estimation has, to the best of our knowledge, never been implemented in a machine learning algorithm used for the specific task of monocular 3D Object Detection. Methods previously used in other applications for extracting uncertainties are applied to the task.

- **Uncertainty evaluation framework** - A framework for comprehensive evaluation of uncertainty estimates is proposed. It constitutes of multiple complementary evaluation metrics, inadequate on their own, but constitute a solid evaluation framework when combined.
- **Global uncertainty propagation** - A framework for propagating uncertainties from the image plane to the global coordinate system is derived in order to present uncertainties completely for all 3D parameters as some parameters are estimated in the image plane.

1.6 Related Work

In the paper by Kendall and Gal [1] they discuss the problem of how today's machine learning algorithms are not capable of understanding their uncertainties. They further discuss and argue that finding a way to describe the uncertainties in a robust way is a relevant problem to solve to make the most out of the machine learning algorithms. Furthermore, they divide uncertainties in machine learning into two sub-categories, epistemic and aleatoric uncertainty.

Epistemic uncertainty describes the uncertainty about our model parameters. For so called out-of-data examples, where the data differs from what the network has seen during training, the epistemic uncertainties increases, which is crucial to know in safety critical applications. It is therefore very important to include because it somewhat enables handling of unknown scenarios by expressing that the estimates are uncertain [1].

Aleatoric uncertainty may be split up into homoscedastic and heteroscedastic uncertainty, the first being a constant uncertainty over the data set and the latter a varying input dependent uncertainty. The homoscedastic uncertainty is a certain static uncertainty for one task, e.g. depth estimation, and another constant uncertainty for another task. Heteroscedastic uncertainty is varying with the input and in the example of depth estimation it is usually low for high resolution, clear images where edges are distinct. The uncertainty is, on the contrary, high for blurry edges, reflections and overexposure [1].

Kendall and Gal further propose the use of Bayesian Deep Learning (BDL) to estimate the uncertainties. Bayesian Deep Learning is a field intersecting deep machine learning and Bayesian statistics which benefits from being able to model epistemic uncertainties. A practical approach to estimating these uncertainties is by using Monte Carlo Dropout as discussed by Nair et al. [2] and Gal et al. [3] where the network is trained with dropout and at test time multiple forward passes are run with different nodes dropped, resulting in multiple predictions that can than be combined to obtain a final estimate. An extension to the traditional dropout is the Concrete Dropout which enables automated tuning of the dropout probabilities which results, according to the authors [4], in a more calibrated [5] model. Gal and Ghahramani [6] describes how to adapt the framework for estimating the epistemic uncertainties through Monte-Carlo Dropout to convolutional neural networks.

Moreover, Deep Ensembles is another approach for approximating Bayesian Neural Networks [7] where multiple networks are trained and their predictions are combined to obtain the final estimate, the idea is that this will reduce the variance in

the predictions and make the model less sensitive to the specifics in the training data as each model will learn slightly different things. A similar approach for describing the uncertainties is with the frequentist approach of bootstrapping [8]. The bootstrapping approach is to train an ensemble of networks where each network is trained with different fractions of the training data, e.g. every networks trains on a unique 67% of the training data. The different predictions are then combined to obtain the final estimate.

In their paper [9] Ilg et al. compare popular uncertainty methods and furthermore introduce a network structure that utilizes the Winner-Takes-All loss to, in a single forward pass, predict multiple hypotheses for optical flow and corresponding uncertainties. The hypotheses are combined to obtain the final optical flow estimates and corresponding uncertainties in a computational efficient way.

Harakeh et. al [10] use BDL to estimate uncertainties in the output for 2D object detectors. They estimate aleatoric uncertainty and extract epistemic uncertainty using Monte-Carlo Dropout as in Kendall and Gal's work. Further, they propose a substitute to non-maximum suppression that is commonly used in object detection to suppress some outputs. Their solution enables prior information to be incorporated in the final estimates. They interpret the output from the machine learning model as measurements of the variables to update the prior distribution to obtain the final conditional posterior distribution for the variables. This way a lot of information, that would normally be thrown away, is instead used to obtain the final estimates.

As described in this section, there are existing solutions that estimates the accompanied uncertainty estimates to machine learning predictions. However, to the best of our knowledge, there is no current solution for estimating uncertainties in monocular 3D object detection nor is there a standardized framework for evaluating uncertainties. The methods used for extracting uncertainties in this thesis is based on the methods presented by Kendall and Gal, Nair et al. and Gal Ghahramani et al. The proposed framework for uncertainty evaluation combines ideas from Kendall and Gal, Ilg et al. and Harakeh et. al.

1.7 Thesis Outline

The remainder of the thesis is structured as follows. Chapter 2 covers the theoretical concepts that this thesis is built upon and presents the task at hand in greater detail. The chapter ends by introducing how the performance of the machine learning method is evaluated and a few metrics for evaluating uncertainties. In Chapter 3, the method describing the conduction of this thesis is presented. The chapter is further divided into first describing the network structure and specifics about choice of parameters and implications thereof. Secondly implementation details such as programming language and parameter settings are presented. Lastly, the evaluation metrics used in the project are discussed. Chapter 4 contains the results that were obtained, which are then discussed in Chapter 5. A conclusion is finally presented in Chapter 6.

2
Theory

2.1 Bayesian deep learning

In Bayesian inference the purpose is to provide a mathematical framework to model systems in such a way that uncertainties of the system are taken into account, to allow for well informed decisions [11]. In Bayesian deep learning the intention is to incorporate Bayesian statistics into deep machine learning. By doing this it is possible to account for, and express, uncertainties in the model. The uncertainties are furthermore often categorized into uncertainty inherent in the model and uncertainty inherent in the data. In other words, they are often categorized into *epistemic* and *aleatoric* uncertainty [1, 12], further explained in Section 2.2.

There are a few fundamental building blocks in Bayesian modelling. These are the prior distribution, the likelihood function and the posterior distribution. The prior distribution $p(\boldsymbol{\omega})$, holds information about prior beliefs or knowledge about the system or parameters. The likelihood function, $p(y|x, \boldsymbol{\omega})$, explains how probable certain outputs, y, are given an input x and a parameter setting $\boldsymbol{\omega}$, where $\boldsymbol{\omega}$ could be a particular set of parameters or weight matrices. The posterior distribution, $p(\boldsymbol{\omega}|x, y)$, is the distribution over the unknown parameters given the data and the distribution that represents the state of knowledge about the parameters when all the information of the model and the observed measurements are used.

In a machine learning setting, the Bayesian mindset is applied by considering the weights in the neural network to be stochastic random variables rather than deterministic values and a prior distribution is placed over each weight. This distribution represents our prior beliefs on the parameters. Given inputs $\mathcal{X} = \{x_1, ..., x_N\}$ and corresponding target outputs $\mathcal{Y} = \{y_1, ..., y_N\}$ we wish to find the posterior distribution over the weights. This posterior distribution captures the set of plausible parameters $\boldsymbol{\omega}$ for a function $y = f(x, \boldsymbol{\omega})$ that are likely to have generated our outputs [1]. The posterior distribution can be calculated using Bayes' rule [11] as

$$p(\boldsymbol{\omega}|\mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y}|\mathcal{X},\boldsymbol{\omega})p(\mathcal{X}, \boldsymbol{\omega})}{p(\mathcal{Y}|\mathcal{X})p(\mathcal{X})}. \tag{2.1}$$

As the weights $\boldsymbol{\omega}$ and the inputs $\mathcal{X}$ are independent of each other without corresponding targets $\mathcal{Y}$, the joint distribution $p(\mathcal{X}, \boldsymbol{\omega})$ can be written as $p(\mathcal{X}, \boldsymbol{\omega}) = p(\mathcal{X})p(\boldsymbol{\omega})$ [1]. This enables the expression for the posterior over the weights to be simplified to

$$p(\boldsymbol{\omega}|\mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y}|\mathcal{X},\boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathcal{Y}|\mathcal{X})}. \tag{2.2}$$

However, the marginal probability

$$p(\mathcal{Y}|\mathcal{X}) = \int p(\mathcal{Y}, \boldsymbol{\omega}|\mathcal{X}) \, \mathrm{d}\boldsymbol{\omega} = \int p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\omega}) p(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega} \tag{2.3}$$

needed to calculate the posterior over the weights in Equation (2.2) can't be evaluated analytically as it requires us to integrate over all possible weight configurations, which is infeasible. Thus the posterior distribution in Equation (2.2) has to be approximated. This can be done by fitting a simple distribution that minimizes the Kullback–Leibler (KL) divergence [13] between the simple distribution and the true posterior distribution over the weights [1, 6] to then use this simple distribution as an approximation of the true posterior. Multiple methods for obtaining this simple distribution exist [6, 14, 15, 16], where the most popular one is a dropout approximation.

To now predict an output, y^*, for a new input point x^* we want to calculate the marginal distribution $p(y^*|x^*, \mathcal{X}, \mathcal{Y})$. This can be done by first identifying the distribution

$$p(y^*, \boldsymbol{\omega}|x^*, \mathcal{X}, \mathcal{Y}) = p(y^*|x^*, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathcal{X}, \mathcal{Y}) \, . \tag{2.4}$$

Integrating over the weights marginalizes the weights out, resulting in

$$p(y^*|x^*, \mathcal{X}, \mathcal{Y}) = \int \underbrace{p(y^*|x^*, \boldsymbol{\omega})}_{\text{likelihood}} \underbrace{p(\boldsymbol{\omega}|\mathcal{X}, \mathcal{Y})}_{\text{posterior}} \, \mathrm{d}\boldsymbol{\omega} \, . \tag{2.5}$$

For regression, Laplacian and Gaussian likelihood functions have been used [1, 7, 9] when approximating the marginal distribution and to model uncertainty inherent in the data. The Gaussian likelihood function can be written as

$$p(y|x, \boldsymbol{\omega}) = N(y; f(x, \boldsymbol{\omega}), \sigma^2)$$

where x is the input to the neural network, $f(x, \boldsymbol{\omega})$ is the output from the neural network with weights $\boldsymbol{\omega}$. This can be thought of as corrupting the output with observation noise.

However, evaluating Equation (2.5) still requires the evaluation of an integral over the weight space, which is explained to be infeasible. A way of approximating this distribution would be to sample from the distribution and to calculate the sufficient statistics of the distribution from the samples. This could, for instance, be done with the popular Monte-Carlo sampling method, Monte-Carlo Dropout [3], that allows i.i.d. samples of the distribution in Equation (2.5) to be drawn by having dropout enabled at test time when running inference with the neural network [1][10]. Through these samples the mean and covariance of the distribution can be approximated as

$$\mu(x_i) = \frac{1}{T} \sum_{t=1}^{T} f(x_i, \boldsymbol{\omega_t}) \tag{2.6}$$

$$\Sigma_e(x_i) = \frac{1}{T} \sum_{t=1}^{T} f(x_i, \boldsymbol{\omega_t}) f(x_i, \boldsymbol{\omega_t})^{\mathrm{T}} - \mu(x_i) \mu(x_i)^{\mathrm{T}} \tag{2.7}$$

where x_i is input number i to the neural network, T is the number of times MC-Dropout sampling is performed, $\boldsymbol{\omega_t}$ is the specific weight configuration after applying dropout at forward pass number t, $f(x_i, \boldsymbol{\omega_t})$ is the neural network regression output for the t^{th} MC-Dropout run. Lastly, $\Sigma_e(x_i)$ models the epistemic uncertainty for the regression variables in the neural network output, further described in Section 2.2.2.

2.2 Uncertainties

To be able to optimally fuse data from different sensors into a single unified estimate some measure of uncertainty for each data source has to be accounted for. The uncertainties for each of the sensors can furthermore be divided into two main categories, aleatoric and epistemic, that differentiates where the uncertainties arises from [12]. These different sources for uncertainties can for instance be imperfections in a model of the world or in variables that are unknown and that therefore can not be accounted for. For instance, if one were to model the trajectory of an apple dropped from an airplane, an imperfection in a model of the world could be to ignore the effect of air resistance and an unknown variable could be the precise wind speed at different altitudes along the trajectory.

2.2.1 Aleatoric Uncertainties

The first category that the uncertainties are divided into are the aleatoric uncertainties. Aleatoric uncertainties arises from noise in the data which could be a blurry or overexposed camera image, dirt on the sensor or occlusion due to various reasons. These uncertainties can further be categorized into *homoscedastic* uncertainties which stays constant for different inputs (for example a constant bias in a sensor) and *heteroscedastic* uncertainty. Heteroscedastic uncertainty depends on the input where some inputs or some part of the input space might have noisier outputs due to occlusion or other similar reasons. For computer vision applications heteroscedastic uncertainties is especially important as the input is normally camera images where different parts of an image might clearly have more uncertain outputs than other. For example, for depth regression where the task is to estimate the distance to each pixel, a highly textured image with distinct edges are expected to have confident predictions. On the contrary, the uncertainty is expected to be high for blurry edges, reflections and overexposed regions.

As aleatoric uncertainty can be thought of as unknown observation noise that is present in the data it can be learned from data. This is achieved by placing a distribution over the output from the model to then try to learn the characteristics that describes this distribution. For example the regression parameters in the output might be modelled as corrupted with Gaussian random noise where we would then want to learn the noise's variance from different inputs [17] as the variance and the mean fully describes a Gaussian distribution.

The Gaussian distribution for output variable y_i can be written as

$$p(y_i|\hat{y}_i) = \frac{1}{\sqrt{2\pi\sigma(x_i)^2}} \exp\left(-\frac{|y_i - \hat{y}_i|^2}{2\sigma(x_i)^2}\right) \tag{2.8}$$

where $\hat{y}_i$ is the mean and $\sigma(x_i)^2$ is the variance of the distribution for input image i. To be able to find an expression for learning the mean and variance for the output variable negative log likelihood is applied as proposed in [1], resulting in

$$\begin{aligned} -\log(p(y_i|\hat{y}_i)) &= -\log\left(\frac{1}{\sqrt{2\pi\sigma(x_i)^2}}\exp\left(-\frac{|y_i-\hat{y}_i|^2}{2\sigma(x_i)^2}\right)\right) = \\ &= -\log\left(\frac{1}{\sqrt{2\pi\sigma(x_i)^2}}\right) - \log\left(\exp\left(-\frac{|y_i-\hat{y}_i|^2}{2\sigma(x_i)^2}\right)\right) = \\ &= \frac{|y_i-\hat{y}_i|^2}{2\sigma(x_i)^2} - \log\left(\frac{1}{\sqrt{2\pi\sigma(x_i)^2}}\right) = \frac{|y_i-\hat{y}_i|^2}{2\sigma(x_i)^2} + \log\left(\sqrt{2\pi\sigma(x_i)^2}\right) \propto \\ &\propto \frac{|y_i-\hat{y}_i|^2}{2\sigma(x_i)^2} + \log\left(\sigma(x_i)\right) \end{aligned} \tag{2.9}$$

where x_i is the input, $\hat{y}_i = f(x_i)$ is the output, y_i is the ground truth regression target and $\sigma(x_i)^2$ is the estimated output variance from the network. Using the final expression in Equation (2.9) the loss for a regression parameter can be written as

$$\mathcal{L}_{\mathcal{G}}(x_i) = \frac{|y_i-\hat{y}_i|^2}{2\sigma(x_i)^2} + \log\Big(\sigma(x_i)\Big). \tag{2.10}$$

The variance of a Laplacian distribution is given by $\sigma^2 = 2b^2$ [18] and in a similar fashion the loss, $\mathcal{L}_{\mathcal{L}}$, for a Laplacian distribution, $p_L(y_i|\hat{y}_i)$, can be written as

$$\begin{aligned} p_L(y_i|\hat{y}_i) &= \frac{1}{2b(x_i)}\exp\left(-\frac{|y_i-\hat{y}_i|}{b(x_i)}\right) = \left\{2b^2 = \sigma^2 \Rightarrow b = \frac{\sigma}{\sqrt{2}}\right\} \\ &= \frac{1}{\sqrt{2}\sigma(x_i)}\exp\left(-\frac{\sqrt{2}|y_i-\hat{y}_i|}{\sigma(x_i)}\right) \Rightarrow -\log(p_L(y_i|\hat{y}_i)) \Rightarrow \\ &\Rightarrow \mathcal{L}_{\mathcal{L}}(x_i) = \frac{\sqrt{2}|y_i-\hat{y}_i|}{\sigma(x_i)} + \log\Big(\sigma(x_i)\Big). \end{aligned} \tag{2.11}$$

The formulation of the loss functions in Equation (2.10) or Equation (2.11) generates the maximum likelihood estimate of the parameters. With these loss functions, the network also has the possibility to attenuate the effect of outliers during training by increasing the estimated variance and thus making the first term in $\mathcal{L}(x_i)$ smaller. The second term acts as a regularizer, preventing the model from rejecting all training examples by always setting the variance to be infinity. The aleatoric uncertainties are thus computationally cheap to model as it only requires the network to estimate twice as many regression parameters and a change of loss function.

2.2.2 Epistemic Uncertainties

The other category of uncertainty is epistemic uncertainty. Epistemic uncertainty, also called model uncertainty, is uncertainty in the output inherent from the uncertainty in the model's parameters. Is our model a good approximation to the function

turning inputs into desired predictions in terms of architecture and weights? As opposed to the aleatoric uncertainties, the epistemic uncertainties can be explained away given more representative training data and it is supposed to increase for out-of-data examples (situations that is different from the training set) [1][3]. In safety critical applications, the ability to be able to identify situations that are different from what the model has seen during training is important to be able to make well informed and balanced decisions. As described in Section 2.1 the epistemic uncertainties can not be calculated analytically but can be approximated with various methods where the most popular one is Monte-Carlo Dropout [1].

2.3 Traditional computer vision

To find objects and recreate scenes [19] from camera images has been done for decades. Traditional methods for, for example, scene reconstruction have practical use cases such as building a map of the surroundings and localization within the map. Such applications often make use of feature extractors, like SIFT [20], to match descriptive regions across images and find the corresponding 3D points that corresponds to the matched regions. Both the matching and finding 3D points can be done in multiple different ways using traditional computer vision algorithms. However, object detection using traditional computer vision techniques is extremely difficult as objects of a specific class that one wishes to detect rarely look the same, which is why machine learning is mostly used for object detection nowadays. In traditional computer vision "hand crafted" features are extracted from the images as opposed to leaving the task of extracting valuable features to a machine learning model. The effect that this has becomes prominent when comparing the results for the ImageNet challenge [21] from before and after 2012 when the first deep machine learning model was used in the challenge.

2.3.1 Projecting 3D points onto the image plane

In this section we will give a brief understanding of how points in images relates to 3D world coordinates through a model of a camera. For more in depth explanations we refer the reader to [22][23].

Given a set of 3D points in homogeneous coordinates $\mathcal{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ and a 3×4 camera matrix P that models the camera, one can derive expressions for the image coordinates, (x, y) that corresponds to the 3D coordinates by

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P\mathcal{X}\,. \tag{2.12}$$

By dividing by λ the image coordinates x, y can be found. In a similar fashion we can derive 3D world coordinates from images coordinates by multiplying Equation (2.12) with the pseudo inverse of the camera matrix $P^\dagger$ on the left, we call this operation a un-projection of a pixel. However, some additional information is required to uniquely determine the world coordinates due to that a pixel in the image could arise

from a ray in world coordinates. This is due to that points in images only have two dimensions whereas points in the world has three dimensions. Therefore, we need some way of determining where to cut the ray in order to find world coordinates from image coordinates. There are multiple ways of handling the depth problem where a common approach is to use a stereo camera, un-projecting two matched pixels and finding the 3D point where the two rays intersects (or the point that minimizes some distance between the rays, if the lines does not intersect in 3D), thus solving the depth problem.

2.4 3D Object Detection

To detect objects in world coordinates is important to be able to interpret and understand the surroundings. This task, called 3D object detection (3DOD), can be done in multiple different ways where natural choices include radar, lidar, stereo cameras or other depth sensors. For depth estimation and 3DOD, modern lidar sensors is a popular choice [24][25]. For a camera to be a viable complement for detecting objects the final estimates has to be in world coordinates (3D) rather than in the image plane for the detections to be useful in other functions. Lately machine learning has been the choice of method for performing object detection. In this section the different components of a machine learning algorithm for 3DOD will be described.

2.4.1 3D Parameters

A 3D bounding box can, assuming that the roll and pitch angles are negligible, be described by its dimensions *H, W, L*, center point T $= t_x, t_y, t_z$ and yaw angle θ. There are multiple different representations of what parameters that one could regress and combine to obtain the parameters that describes a 3D bounding box. One can for instance regress the final representation straight away as done in [26, 27] or regress parts of the final parameters and derive the remaining ones from geometric constraints as in [28].

2.4.2 Network outputs

In this section a parameter representations for when the 3D regression parameters are not estimated directly will be described. The geometrical constraints that relates the output parameters to the final variables will further be described in Section 2.4.5.

As discussed in [28] the global orientation or the yaw angle (rotation around world coordinate Y axis) of an object is hard to estimate directly due to that an object that has a constant global orientation might appear very differently in the image plane depending on where in the image it is located. However, the observation angle, α, from which the camera views the object is easier to estimate as that angle is closer to the raw data. A visualization of how the observation angle changes as an object with a constant yaw angle passes by the camera can be seen in Figure 2.1.

The variance of the dimension estimate is usually quite small (cars tend to be roughly the same size), making it suitable to estimate the dimension variable di-

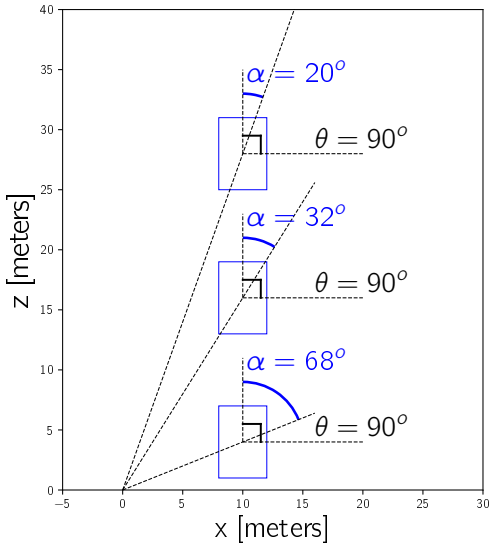


Figure 2.1: An illustration of how the observation angle α changes when an object with a constant yaw angle θ passes by the camera.

rectly. Furthermore, these variables stay constant as the objects moves or rotates in relation to the camera. A possible choice could be to estimate the logarithm of the dimension variables in order to solve the problem of negative dimensions. However, as the outputs are assumed to be a distribution this would require the estimated uncertainties to be transformed from uncertainties in the logarithm of the dimension to uncertainties in dimension, $\sigma_{\log(d)} \rightarrow \sigma_d$.

In this thesis the only sensor data that is used is images from a single camera. As no depth information is available and since monocular reconstruction suffers from scale ambiguity some depth estimate has to be regressed to be able to resolve this problem. There are again different representations that can be chosen for the depth variable. For instance, the distance along the Z axis, (the longitudinal distance) or the euclidean distance to the objects can be chosen as a regression parameter. The two depths are illustrated in Figure 2.2 where three objects has the same Z depth to the camera whilst the euclidean distance differs.

Lastly the 2D bounding box, that encloses the object in the image plane, is typically also estimated to be able to use it in a scoring function but also for deriving world coordinates from it, further described in Section 2.4.5. In addition to the regression variables a classification estimate for the predetermined classes has to be included for the network to be able to classify the objects in the image.

2.4.3 Pixel-wise estimations

Similarly to what parameters to regress with the neural network there are also multiple ways of choosing the output resolution from the neural network. Some approaches includes using only a few locations in the image (also called anchor points) [29, 30], for the network to output estimates at. Another approach is to output estimates at a much higher resolution, as done in [31]. During training the outputs of higher resolution are connected through support regions where each pixel

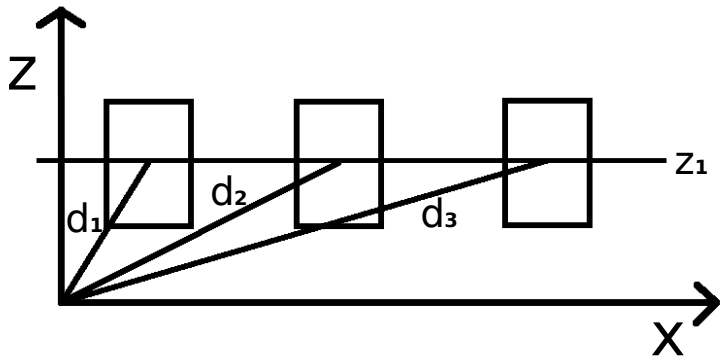


Figure 2.2: Illustration of different depth representations to be able to resolve the scale ambiguity that monocular reconstruction suffers from. The euclidean distances d_1, d_2 and d_3 differ but the distance along the Z axis is the same for the three objects.

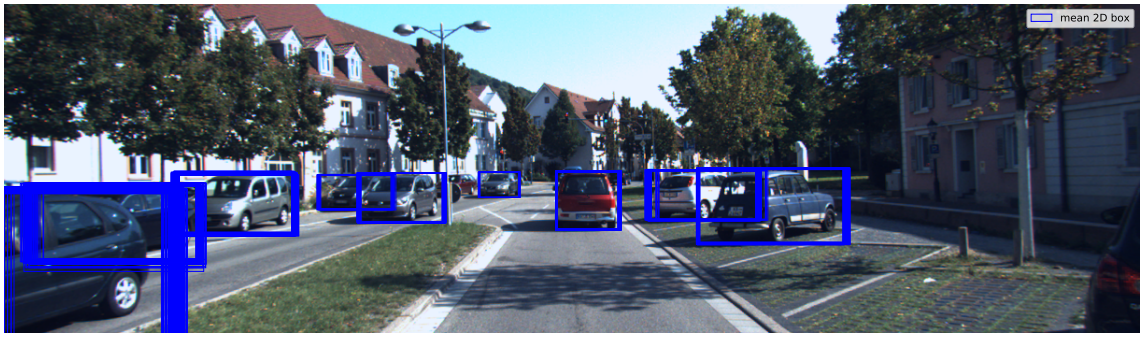


Figure 2.3: Output 2D bounding boxes after an initial classification threshold has been applied. Each object has multiple estimates for its parameters.

in the region holds the regression targets and the classification one-hot target vector for the object. The Support region for an object is created in the center of the object's ground truth 2D bounding box with its height and width being 20% of the ground truth box height and width, respectively. With a high resolution output map, a way of removing false detections and clustering estimates that are likely to corresponds to the same object is needed. As a first step to remove estimates that are likely to not be correct and correspond to an actual object a classification threshold is applied. In this step all the output pixels that do not have a classification score (softmax of logit vector) over a certain threshold for any of the objects that we have trained the network to detect will be removed, ideally removing everything but the duplicate detections where there is an object. This could result in a output as the one in Figure 2.3 where the detected objects have multiple estimates for their parameters. Thus further processing of the output is required before ending up with the final estimates.

2.4.4 Box-selection through Non-Maximum-Suppression

After all of the post processing steps have been done we want *one* bounding box per object in the frame. With the chosen output resolution there will be multiple

Figure 2.4: Output 2D bounding boxes when boxes have been suppresed by the NMS algorithm.

boxes that all originates from the same object (see Figure 2.3) and thus we have to find a way of clustering these together. This is done by applying a Non-Maximum-Suppression (NMS) algorithm that suppresses outputs based on their 2D bounding box overlap with other 2D bounding boxes of the same class. The NMS algorithm is deployed once for each class in the classification head.

Given a set of estimates the NMS algorithm begins with picking the output estimate that has the highest classification score (softmax of logit vector). This estimate is saved and the 2D IoU for this box with all of the other outputted 2D bounding boxes is calculated. If any other box (of the same class) has a 2D IoU over a certain threshold value the output is removed and the procedure restarts with whatever output estimate that has the highest classification score of the remaining outputs. This is repeated until only boxes that does not have 2D IoU above the chosen threshold remains.

When the algorithm has finished there will (ideally) be one output estimate for each object. What the algorithm effectively does can be seen by comparing Figure 2.4, where the NMS algorithm has run, with Figure 2.3 that was the input to the algorithm. Note that, in Figure 2.4, all of the detections for the second most leftward car in the image is suppressed as its 2D bounding box has an IoU above the chosen NMS threshold with the most leftward car.

2.4.5 Derive 3D parameters from 2D estimates

As described in Section 2.4.2 all 3D parameters are not necessarily regressed from the neural network and thus the parameters that are not have to be derived as a post processing step. The 3D bounding box parameters that are not regressed and thus has to be derived are the X and Y components of the object's 3D center, t_x and t_y and the yaw angle θ.

To find the object's 3D location we start by finding the object's center in the image plane that is found by simply computing the mean of the regressed 2D bounding box coordinates $\mathbf{c} = (x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}})$. That is

$$c_{\text{center}} = \frac{1}{2} \begin{bmatrix} x_{\text{max}} - x_{\text{min}} \\ y_{\text{max}} - y_{\text{min}} \end{bmatrix} . \tag{2.13}$$

The camera matrix, explained in Section 2.3.1, is known, thus we can find a ray

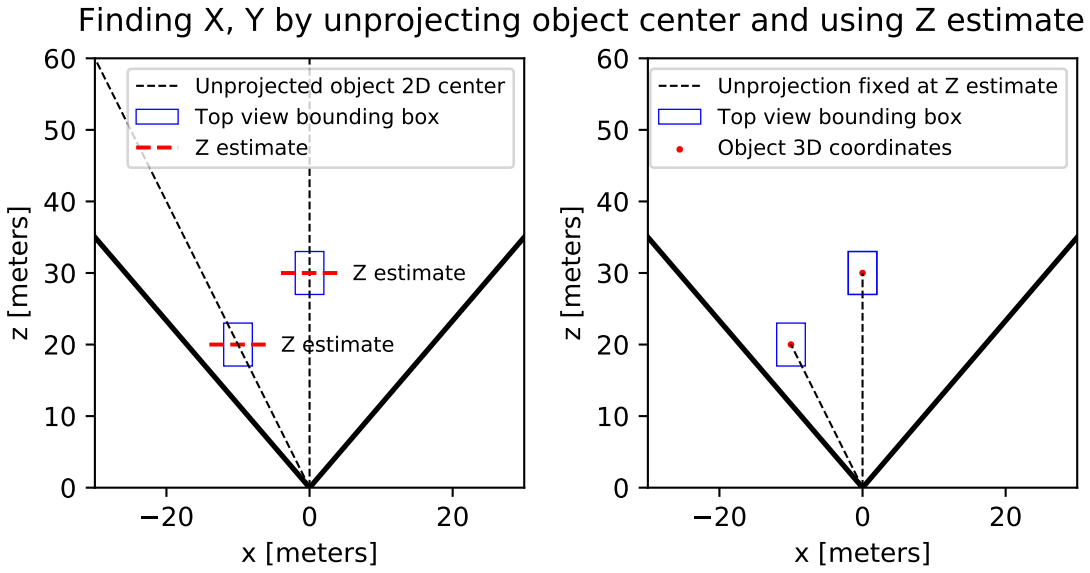


Figure 2.5: Top view illustration of how the object center is unprojected into world coordinates and how the Z estimate from the neural network is used to solve the problem that a pixel in an image corresponds to a line in world coordinates. Note that this illustration only shows the X component but the lines are actually in 3D and thus Y is obtained simultaneously.

from the camera center for which the object's 3D center point, or 3D location, lies on by unprojecting (inverting Equation (2.12)) the object's center, c_{center}, in the image plane.

We know that the scale, s, is arbitrary when unprojecting a single pixel into world coordinates. The ray, T^*, for which the object's 3D location, T, lies on is found by

$$P^{\dagger}\begin{bmatrix} c_{center} \\ 1 \end{bmatrix} = s\begin{pmatrix} t_x \\ t_y \\ t_z \\ 1 \end{pmatrix} \Rightarrow \; T^* = s\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = \begin{pmatrix} t_x^* \\ t_y^* \\ t_z^* \end{pmatrix}. \tag{2.14}$$

The object's 3D location T on the ray T^* can be found by scaling the vector T^* with a factor such that the third component of T^* equals the estimated Z depth, Z_{nn}, where Z_{nn} is the estimated distance along the Z axis to the object's center, yielding

$$T = \frac{z_{nn}}{t_z^*}T^* = \begin{pmatrix} t_x \\ t_y \\ z_{nn} \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}. \tag{2.15}$$

The procedure for finding the X and Y world coordinate is furthermore illustrated in Figure 2.5.

The final parameter that has to be derived is the global orientation or the yaw angle, θ, for which the object is rotated around the Y axis with. Given the observation angle, α, and the X, Z coordinates from the object's 3D location the yaw angle, θ can be calculated as

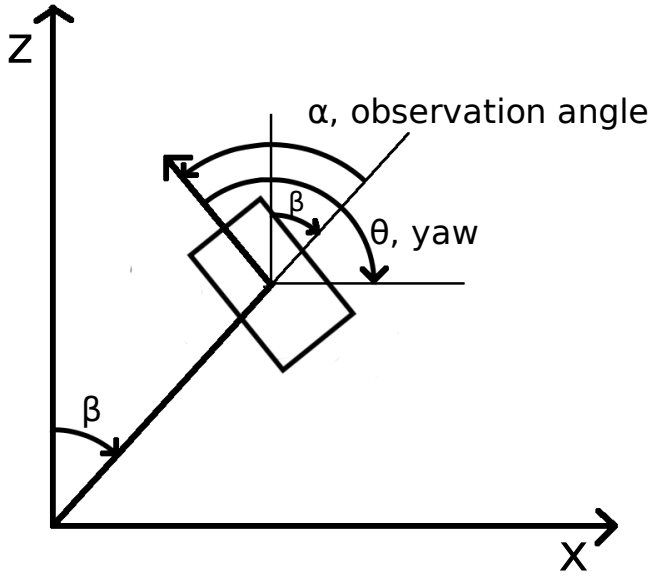


Figure 2.6: Illustration of the different angles that is used to calculate the global orientation (yaw angle, θ) using the observation angle α and the angle $\beta = \arctan\left(\frac{X}{Z}\right)$.

$$\theta = \alpha + \arctan\left(\frac{X}{Z}\right). \tag{2.16}$$

Each respective angle from Equation (2.16) is visualized in a top view example in Figure 2.6.

2.5 Performance evaluation

To compare trained networks of different architecture, appropriate evaluation methods have to be found. As the original network architecture, the baseline network, will be altered and re-trained to also include uncertainty estimates it is not unlikely that the pure performance of the network will differ from the baseline network. This makes it interesting not only to have metrics to evaluate how good the uncertainties estimates are but also what performance the network has.

2.5.1 KITTI benchmark

In the KITTI benchmark [32] different methods for 3D Object Detection on the KITTI dataset [33] can be compared in terms of their pure performance. The performance metric used in this benchmark is the average precision (see Section 2.5.5), using the PASCAL criteria [34]. This benchmark has three different difficulties where different filters are used to filter out objects before calculating the average precision. The filters make sure that e.g. objects that are too far away, object that

Table 2.1: The three different difficulties and each respective filter for the difficulty used in the KITTI benchmark.

Difficulty	Min. bounding box height	Max. occlusion level	Max. truncation	Car bounding box overlap	Pedestrians and cyclists bounding box overlap
Easy	40 Px	Fully visible	15 %	70 %	50 %
Moderate	25 Px	Partly occluded	30 %	70 %	50 %
Hard	25 Px	Difficult to see	50 %	70 %	50 %

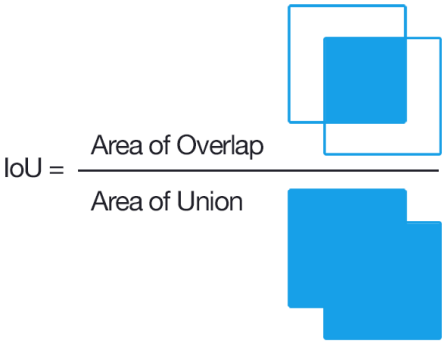


Figure 2.7: A visual equation for Intersection over Union (Jaccard Index) [35].

are too truncated or occluded are not accounted for. The filters for the different difficulties are summarized in Table 2.1.

2.5.2 Intersection over union

Intersection over union, also known as the Jaccard index, is a performance metric evaluating the similarity and diversity of two sets, in object detection this would be two areas or volumes. The Jaccard index is calculated by dividing the size of the intersection with the size of the union of the two sets, as in Equation 2.17.

$$\mathrm{IoU}(A, B) = \frac{A \cap B}{A \cup B} \tag{2.17}$$

This is furthermore illustrated in Figure 2.7.

In object detection the separation of true positives from false positives is often determined based on a detection's IoU in comparison to ground truth objects. This comparison can be done in multiple different dimensions where the most common ones are

- 2D IoU - Bounding box in the image plane,
- 3D IoU - Bounding box in world coordinates,
- IoU from bird's-eye-view (BEV) - Bounding box in top view.

Furthermore, the mean IoU for each of these dimensions over the whole dataset can be used to indicate performance of a model.

2.5.3 Detection statistics

Detections statistics is a performance metric keeping track of the number of different detections or the absence of them. A detection is here defined as having a

classification confidence above a certain threshold. A detection is assigned as a true positive to an annotated object if a scoring function for the two objects is above a given threshold, a typical scoring function is IoU in some dimension. A split refers to a certain filtering of objects that does or does not satisfy some constraint, for example, only include objects that has a minimum height of 40 pixels.

- **True Positive (TP) -** The number of detections inside the split that were assigned to ground truth objects.
- **False Positive (FP) -** The number of detections inside the split that were not assigned as true positives to a ground truth object or to an object annotated as non descriptive object.
- **Don't care -** The number of detections inside the split that were assigned as true positives to non descript objects.
- **Duplicate detections -** The number of detections inside the split that are assigned to ground truth objects as duplicate detecitons.
- **Predicted positives -** All detections made. This is thus the summation of the number of true and false positives, don't care detections and detections being filtered out due to not fulfilling the current split constraints.
- **Non descripts -** The number of objects that is annotated as non descripts due to lack of ground truth data for them.
- **False Negatives (FN) -** The number of objects inside the split that were not assigned to any detection.

2.5.4 Precision, recall and F1 score

Precision and recall are two performance metrics evaluating the ability to detect all real objects and not to make detections for non existing objects. In object detection a detection is classified as a TP if the detection got a score from a scoring function above a predetermined threshold value. An object can thus change from being a TP to a FP by merely changing the scoring function. The three most commonly used scoring functions are 2D, 3D and BEV IoU.

Precision and recall are metrics describing the fraction of the outputted detections that are actually TPs and the fraction of all objects that were detected, given a specific scoring function. Precision is calculated by dividing the number of true positives with the number of true and false positives, thus describing the fraction of the outputs that are assigned to a ground truth object (see Equation 2.18). Recall is the rate of which objects are detected, i.e. number of true positives divided by number of ground truth objects (see Equation 2.19). Precision and recall are illustrated in Figure 2.8. F_1 score is the harmonic mean of the precision and recall and reaches its best value at 1 and lowest value at 0 and is calculated as in Equation 2.20.

$$Precision = \frac{TP}{TP + FP} \tag{2.18}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.19}$$

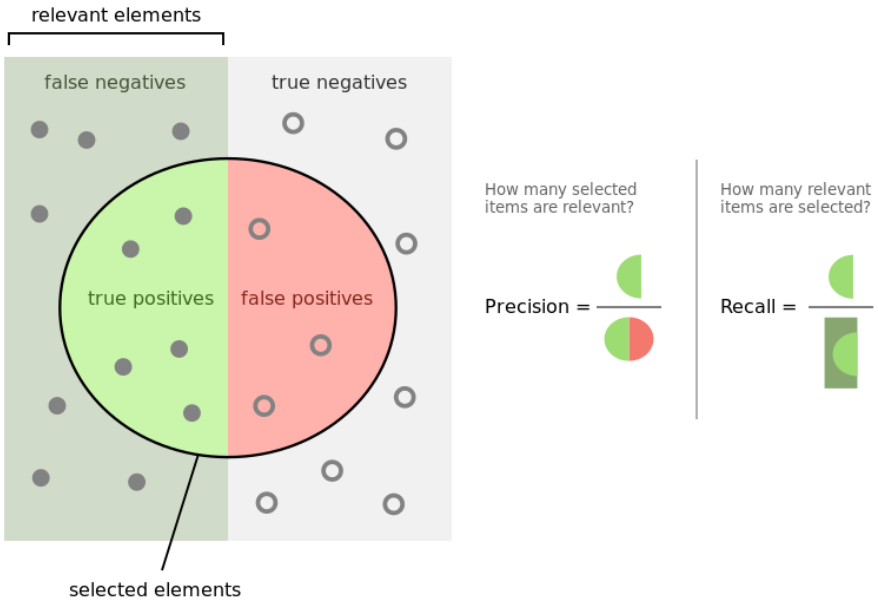


Figure 2.8: Illustration of precision and recall that are central metrics in evaluating performance of 3DOD algorithms [36].

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.20}$$

2.5.5 Average Precision

Average Precision (AP) is a perfomance metric combining combining precision and recall while varying the classification threshold. AP is defined as the area under the precision-recall curve and evaluates precision and recall in relation to each other. The precision-recall curve is created by choosing different classification thresholds for what detections that are accepted and for each different threshold registering the precision and recall for that threshold. As this approach will not give us a continuous precision-recall curve a way of interpolating the curve to be able to compute the area under it is required. There are several common methods for calculating AP (interpolating the curve) but the most common for object detection benchmarking is called voc07, where voc is short for visual object classes and related to the PASCAL criteria [34]. This metric is used when comparing models in the KITTI object detection benchmark described in Subsection 2.5.1. Typical for the AP_{voc07} is its evenly spaced recall levels {0, 0.1, ... 1.0} and it is calculated as

$$AP_{\text{voc07}} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} p_{interp.}(r) \tag{2.21}$$

where $p_{interp.}(r)$ is an interpolated precision taking the maximum precision over all recalls greater than r:

$$p_{interp.}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}). \tag{2.22}$$

Average precision reaches its best value at 1, when the precision is 100 % for 100% recall, and lowest value at 0, when the precision is 0 % for all rates of recall.

2.6 Uncertainty evaluation

How one evaluates the quality and correctness of the uncertainty estimates is not obvious as there is no ground truth uncertainty data to compare to. There are, however, other methods for evaluating uncertainties. One way is to analyze if detections with high uncertainties also have larger errors compared to detections with low uncertainties, i.e. analyze if the accompanied uncertainties are ordered correctly [9]. A second way of evaluating uncertainty estimates is to analyze if the estimated uncertainties coincides with the assumed distribution, described in Section 2.2.1. In other words, analyzing if the parameters seem to follow the assumed distribution, characterized by the estimated uncertainty and the mean estimate. A third way is by analyzing if the uncertainties can help us distinguish true positives from false positives. That is, analyzing if the uncertainties for false positives are consistently higher than for true positives. In the following sections metrics that evaluates these, sought after, properties for the uncertainties will be described.

2.6.1 Area under sparcification error (AUSE)

This metric is used to compare uncertainty estimates for multiple detections to each other and how they relate in terms of magnitude. One would hope that the network indicates large uncertainties where the actual error (compared to the ground truth) is large and small uncertainties where the actual error is small. Thus estimates that are sorted in order of decreasing uncertainty estimates should hopefully also be sorted in order of decreasing error compared to the ground truth.

To calculate the AUSE there are a few steps that has to be done. Firstly, the sparcification curve is created by sequentially removing the estimates that the network indicates the highest uncertainty about and calculating an arbitrary performance metric for the remaining estimates as estimates are removed. The sparcification curve has fraction of removed estimates on the x axis and average arbitrary performance metric on y axis. If the relative order of the uncertainty estimates is good the sparcification curve should be monotonically decreasing when removing the estimates that the network is the most uncertain about. This should retain the good estimates, that should have a low error. If the sparcification curve remains flat then the *order* of the uncertainty estimates are no better than random values and if the derivative of the sparcification curve is positive then the *order* of the uncertainty error are even worse than random, i.e. the uncertainty estimates are rather sorted in reversed order. The sparcification curve can be compared to an oracle curve, created by removing the estimates that, compared to the ground truth, actually have the largest error, sequentially. The difference between these two curves is called the Sparcification Error (SE) curve. This will create a curve with the SE on the Y axis and fraction of removed estimates on the X axis. The area under this SE curve (AUSE) can be used as a single value evaluation metric describing the relative correctness of the uncertainty estimates [9]. As different regression variables and

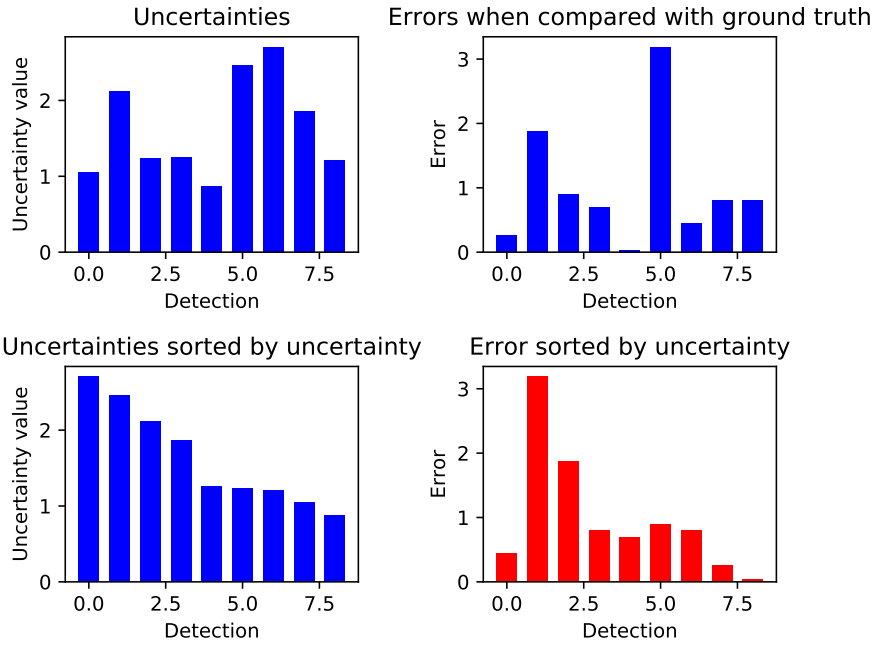


Figure 2.9: The sorting phase of the sparcification curve generation. Estimated uncertainties (upper left) is sorted by their value ranging from highest to lowest (lower left). The actual errors when compared to a ground truth target (upper right) are also sorted in an equivalent order as the uncertainties resulting in a not necessarily optimal sorting for the errors (lower right).

different models all have different oracles a comparison using a single sparcification plot is not possible. However, since the sparcification *error* curve has normalized the oracle out a fair comparison across variables and models can be done with a SE curve or the AUSE metric. Best possible AUSE score is 0 which means that the relative order between the uncertainty estimates coincides with the actual error to the ground truth. This is visually explained in Figure 2.9 and Figure 2.10.

2.6.2 Calibration plots

To be able to trust the output from a neural network we want it to not be under nor over confident. Pose that a neural network is trained to classify images of dogs and cats. Given 10 images where the network has classified each image to be 70% dog and 30% cat we want 7 out of those 10 samples to be an image of a dog and 3 of the samples to be an image of a cat. If the frequency of occurrence equals the outputted probability for the whole range of probabilities (0% to 100%) the network is said to be perfectly *calibrated* [5].

For the regression parameters similar reasoning can be applied as the regression parameters are assumed to be densities rather than point estimates. The probability density function for each regression parameter can be integrated over to find ranges that the ground truth estimate should, analytically, lie within with a certain probability. In a similar fashion to that of the classification case this can be done for probabilities from 0 to 100 percent and the frequency of occurrence can be compared

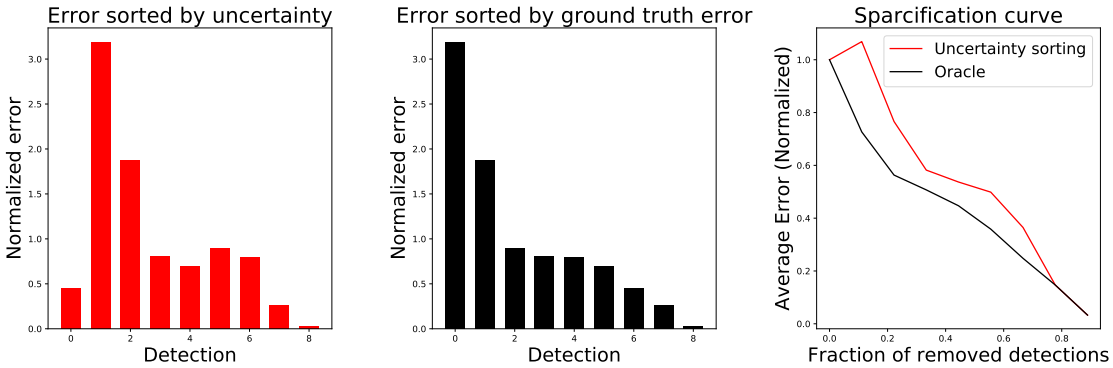


Figure 2.10: The sparcification curve (right) is created by sequentially removing the detections in the order that they were sorted and calculating the average error (normalized by the initial average error) for the remaining samples. This is done for detections sorted by uncertainties (left) and detections sorted by actual error to ground truth (middle).

to the probability of which it should occur.

Given a Gaussian distribution with mean μ and variance σ^2 the probability, c, for a sample from the distribution to lie within a symmetric interval around the mean value can be calculated by

$$c = \int_{\mu-x_1}^{\mu+x_1} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x\,. \tag{2.23}$$

Similarly, the probability for a sample from a Laplacian distribution with mean μ and variance $2b^2$ [18] to lie within a symmetric interval around the mean can be calculated as

$$c = \int_{\mu-x_1}^{\mu+x_1} \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right) \mathrm{d}x\,. \tag{2.24}$$

By now changing $x_1 > 0$ so that c ranges from 0 to 100 percent and continuously registering if the ground truth target for the regression parameter lies within the range $\mu - x_1$ to $\mu + x_1$ the calibration for the regression parameters can be evaluated. This is done by comparing the frequency for which the ground truth target lies within some certain interval with the analytic probability that it should lie within this interval. As this requires a ground truth target to be available for each of the regression parameters this can only be done for detections classified as TPs. Furthermore, the Mean Squared Error to perfect calibration (y = x) is calculated for each parameter to be able to compare calibration across models and parameters. A (perfectly) calibrated network is very useful as that indicates that the outputs from the network can accurately be described by a distribution rather than point estimates.

2.6.3 Minimum Uncertainty Error (MUE)

In object detection an estimate is usually accepted as a detection if the probability assigned to the estimate in the classification head is higher than some threshold. When further analyzing the detections they can be divided into True Positives (TPs) and False Positives (FPs) depending on if they meet some criterion (normally Intersection over Union, IoU, in some space) to a ground truth box of the same class. MUE is a metric describing the ability to discriminate TPs from FPs based on a detections uncertatinties [10][37].

For every TP and FP the categorical entropy and the entropy for the assumed density (Gaussian or Laplacian) is calculated. The categorical entropy indicates how confident the network are about the class for the detection and can be calculated as

$$H_{categorical} = -\sum_i p_i \log p_i \,. \tag{2.25}$$

If the network indicates complete certainty about a class the probability for that class will be 1 and the probability for the other classes will be 0 which will result in $H_{categorical} = 0$. On the contrary the categorical entropy is maximized if the network indicates equal probability for all of the classes.

For the regression parameters a specific distribution is assumed as described in Section 2.2.1. For the case of a Multivariate Gaussian distribution the entropy, H_{Gauss}, can be calculated as

$$H_{Gauss} = \frac{D}{2}\Big(\log(2\pi) + 1\Big) + \frac{1}{2}\log|\Sigma| \tag{2.26}$$

where D is the dimension and Σ is the covariance matrix for the multivariate Gaussian distribution. If the variables are independent, the entropy can be written as

$$H_{Gauss} = \frac{D}{2}\Big(\log(2\pi) + 1\Big) + \frac{1}{2}\sum_{j=1}^{D}\log\sigma_j^2 \tag{2.27}$$

where σ_j^2 is the variance for Gaussian variable j. Similarly, the entropy for multiple, assumed independent, Laplacian variables can be calculated as the sum of the individual entropies as

$$\begin{aligned} H_{Laplace} &= \sum_j \log(2b_j e) = \left\{2b^2 = \sigma^2 \Rightarrow b = \frac{\sigma}{\sqrt{2}}\right\} = \\ &= \sum_{j=1}^{D}\Big(\log(\sigma_j) + \log(\sqrt{2}) + 1\Big) = D\left(\log\left(\sqrt{2}\right) + 1\right) + \sum_{j=1}^{D}\log(\sigma_j) \end{aligned} \tag{2.28}$$

where σ_j is the standard deviation for Laplace variable j. Note that the entropy can be calculated for all of the regression parameters jointly or a chosen subset of them. A high entropy corresponds to high uncertainty among the included regression parameters.

The entropy of the uncertainty estimates for the regression parameters is calculated as in Equation (2.27) or (2.28), depending on the assumed density for the regression variables and the categorical entropy of the classification probabilities is

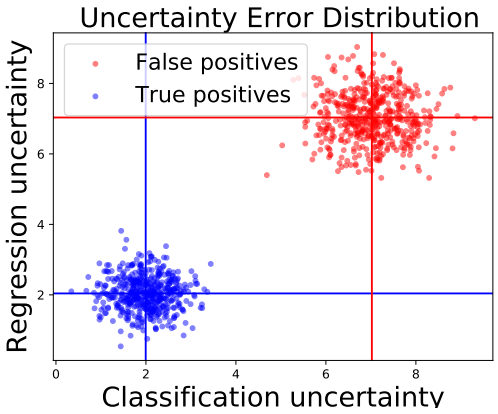


Figure 2.11: An example uncertainty error plot. The blue and red lines mark the average entropy for the true and false positives respectively.

calculated as in Equation (2.25) for each and every one of the TPs and FPs. Comparing the entropies for the TPs and FPs to a threshold value the Uncertainty Error (UE) can be calculated as

$$UE(\delta) = 0.5\frac{|TP > \delta|}{|TP|} + 0.5\frac{|FP \leq \delta|}{|FP|} \tag{2.29}$$

where δ is a specific value for the uncertainty measure (entropy) threshold. This is calculated for the categorical entropy and the regression entropy separately. MUE is defined as the lowest achievable UE when changing δ. In other words MUE for each entropy is achieved at the δ that best separates the TPs from the FPs.

If the regression entropy for the chosen regression parameters and/or the categorical entropy of the classification probabilities for all of the TPs are consistently lower than that of the FPs the uncertainties can be used to further discriminate TPs from FPs. The best possible MUE is 0 where the TPs and FPs are fully separated. A MUE value of 0.5 is the highest (worst) possible MUE. A mean entropy for TPs below the mean entropy for FPs indicates that the uncertainties has the right tendency, and can still be valuable even if the TPs and FPs are not fully separable. An example plot of uncertainty error can be seen in Figure 2.11 where the TPs are visualized in blue and the FPs in red at the values of their classification and regression entropies.

3
Method

In this section the work flow will be described. Firstly, the baseline network and our extensions to include uncertainties will be gone through. The chosen regression parameters, how the 3D variables are derived and how the uncertainties are propagated will be presented. Implementation details will be specified and lastly the process of evaluating the results will be gone through.

3.1 Network structure

Information regarding the baseline network, that the project extends, will be presented. Our solution and extension of this baseline network will then be presented. The network extended with uncertainty estimates is hereby called UncertaintyNet.

3.1.1 Baseline network

The base network structure is adopted from [31][38]. The input to the network is a single image which is passed through a CNN encoder, more precisely the dilated ResNet *drn_c_26* [39]. The last two fully connected layers are removed resulting in the image being down sampled by a factor of 8, resulting in a dense feature vector of dimension 512. This feature vector is passed through separate task specific decoders called task nets, each consisting of two 1 x 1 convolutional layers and a tiling upsampling, resulting in an output that is one fourth of the input image resolution. The output from the task nets is the final network output which is then post processed to yield 3D boxes. The baseline network can be seen in Figure 3.1.

During training of the baseline network, the simple L1 loss, which penalizes the distance between the prediction and the ground truth, is used,

$$L_1 = \sum_{i=1}^{n} |y_i - f(x_i)| \,,$$

where y_i is the ground truth value, $f(x_i)$ the output predictions and n is the number of output parameters. The baseline network will be used during evaluation for comparing performance results.

3.1.2 Our network - UncertaintyNet

The model in this project is based on the baseline model described in Section 3.1.1. The main extension of this network structure is that it includes also aleatoric uncer-

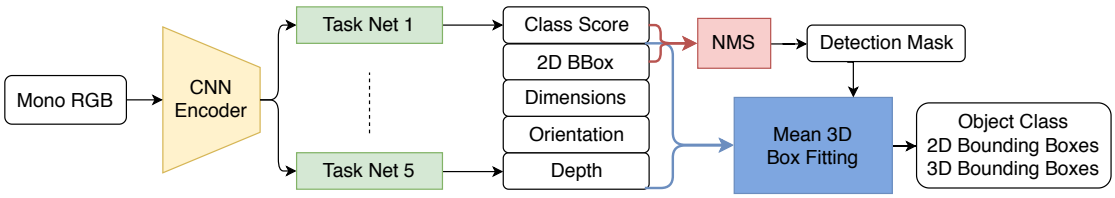


Figure 3.1: The baseline network used to compare performance with. The classification score and the 2D bounding boxes from the output is passed to a Non-Maximum-Suppression algorithm to form a detection mask. For each detection passed by the detection mask a 3D box is found using the network outputs. Image source: [38].

tainties, thus an extra variable σ, for each mean parameter is estimated. Furthermore our network can also estimate the epistemic uncertainties by running multiple forward passes with dropout enabled and calculating the predictive variance of the outputs. The variables that the network estimates are:

- Class Score
- 2D Bounding Box
- Standard deviation for 2D Bounding Box
- Depth
- Standard deviation for Depth
- Orientation
- Standard deviation for Orientation
- Dimensions
- Standard deviation for Dimensions

All of our experiments have been performed with the assumption that the network output is corrupted with observation noise from either a Gaussian or a Laplacian distribution. Heteroscedastic aleatoric uncertainties, where the observation noise is input dependant, and thus can vary across outputs, has mainly been used. The mean estimates and the aleatoric uncertainties are learned by performing backpropagation on the loss function. Depending on the assumed distribution one of two loss functions has been used. For the models trained under the assumption of Gaussian observation noise the loss function in Equation (2.10) has been used and similarly for the models trained under the assumption of Laplacian observation noise the loss function in Equation (2.11) has been used, as the loss function is what defines the parameters to be from a certain distribution.

Furthermore, as there is only ground truth annotations for the regression variables in the generated support regions (SR), discussed in Section 2.4.3, only the estimates in these regions should generate a loss to perform backpropagation on. However, the ground truth masks are constructed to have the same resolution as the network

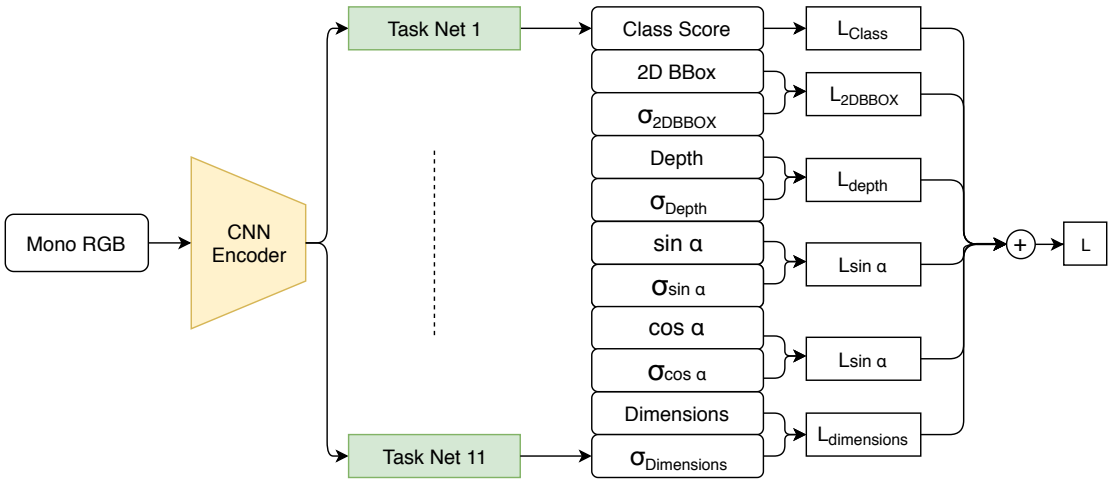


Figure 3.2: The network structure for training. All parameters and their corresponding uncertainties form an individual loss term. The individual loss terms are summarized to form the final loss for which backpropagation is performed on.

outputs but filled with zeros outside support regions. Therefore the network output is masked out (set to zero) outside the support regions, effectively making sure that the difference to the ground truth mask is zero and thus the gradient from these pixels will be zero.

When the network output has passed the suppression mask, the loss for e.g. the depth estimates given an input image x under the assumption of Laplacian output noise is calculated for each pixel $j \in SR$ as

$$L_{depth}(x) = \sum_{j} \frac{\sqrt{2}|y^{GT}_{j,depth} - \hat{y}_{j,depth}|}{\sigma_{j,depth}(x)} + log(\sigma_{j,depth}(x))$$

where $\hat{y}_{j,depth}$ is the estimate of the depth, $y^{GT}_{j,depth}$ is the ground truth regression target and $\sigma_{j,depth}(x)$ is the estimated uncertainty at pixel j and where j belongs to the pixels in the support regions.

The classification head, which has the background class as target outside the support regions, is trained also outside the support regions. A standard cross entropy loss function is used for the classification head. The total loss for a forward pass is the sum of all task specific losses as illustrated in Figure 3.2.

To obtain the epistemic uncertainty estimates we want to calculate the marginal distribution $p(y^*|x^*, X, Y)$ in Equation (2.5). As mentioned in Section 2.1 this can not be calculated analytically but we can draw samples from the distribution in various ways. The method that is used to sample from this is the Monte-Carlo sampling method, Monte-Carlo Dropout [3], that allows us to draw samples from the distribution by having dropout enabled at inference time, running T forward passes and calculating the mean and variance as

$$\mu(x_i) = \frac{1}{T}\sum_{t=1}^{T} f(x_i, \boldsymbol{\omega_t}) \tag{3.1}$$

$$\Sigma_e(x_i) = \frac{1}{T}\sum_{t=1}^{T} f(x_i, \boldsymbol{\omega_t}) f(x_i, \boldsymbol{\omega_t})^T - \mu(x_i)\mu(x_i)^T \tag{3.2}$$

where $\boldsymbol{\omega_t}$ is the network parameters and $f(x_i, \boldsymbol{\omega_t})$ is the network output for forward pass t. Implementation-wise this requires the network to have a dropout layer after every weight layer in the network [6]. We chose to only have dropout in the encoder part of the network as the decoders are very brief (two 1x1 convolutional layers) compared to the encoder. Training is done as usual with active dropout layers. This does not increase the training time for a model. During inference dropout layers remain active, which usually is not the case, and multiple forward passes are performed with the same input image. Multiple forward passes increases the total number of calculations required by the same factor as number of forward passes performed.

An important thing to note here is that when performing multiple forward passes through the network the mean value for the output's distribution is sampled to calculate the epistemic uncertainties. According to the Central Limit Theorem, when sampling from *any* distribution the sampling distribution of the mean values approaches a normal distribution as the number of samples goes to infinity [40]. Thus the predictive variance (epistemic uncertainty), should approach the variance of a normal distribution as the number of forward passes (samples) approaches infinity. In the case where Laplacian observation noise has been assumed this is disregarded and the combined uncertainty, $\sigma^2_{al+ep} = \sigma^2_{aleatoric} + \sigma^2_{epistemic}$, is approximated as the variance for a Laplacian distribution. However, the analysis of the uncertainties for each model has been done for the uncertainties separately as well as combined.

During inference the class score and the 2D bounding box outputs are used together in a Non Maximum Supression (NMS) algorithm to generate a detection mask filtering out unique (see Section 2.4.4) detections with classification confidence higher than a certain threshold. Mean 3D bounding boxes are generated from the mean 2D parameters and NMS detection mask via 3D box fitting, as described in Section 2.4.5. The 3D uncertainties are propagated from 2D to 3D by passing the 2D bounding box, depth and uncertainty estimates along with the detection mask through an uncertainty propagation algorithm, which is described in Section 3.1.4. The complete flowchart is visualized in Figure 3.3.

We will now specify the chosen network outputs and potential encodings of the parameters mentioned in Section 2.4.2. The class scores are estimated as a regular classification task where the available classes are Background, Car, Pedestrian and Cyclist. The 2D Bounding Box coordinates, $(x_{\min}, y_{\min}, x_{\max}, x_{\max})$ are encoded and learned as relative coordinates, i.e. for every pixel (x, y) in the support region the pixel distance from that pixel to the objects boundaries is estimated as

$$f(2Dbbox) = (x - x_{\min}, y - y_{\min}, x_{\max} - x, y_{\max} - y). \tag{3.3}$$

The depth to the object is estimated in meters and is chosen as the distance to an object along the Z axis. One could also choose to estimate the euclidean distance to the object but as all parameters are assumed to be independent it is tractable to keep the variables separated as much as possible. The euclidean distance, and the corresponding uncertainty in euclidean distance, would be used to calculate the

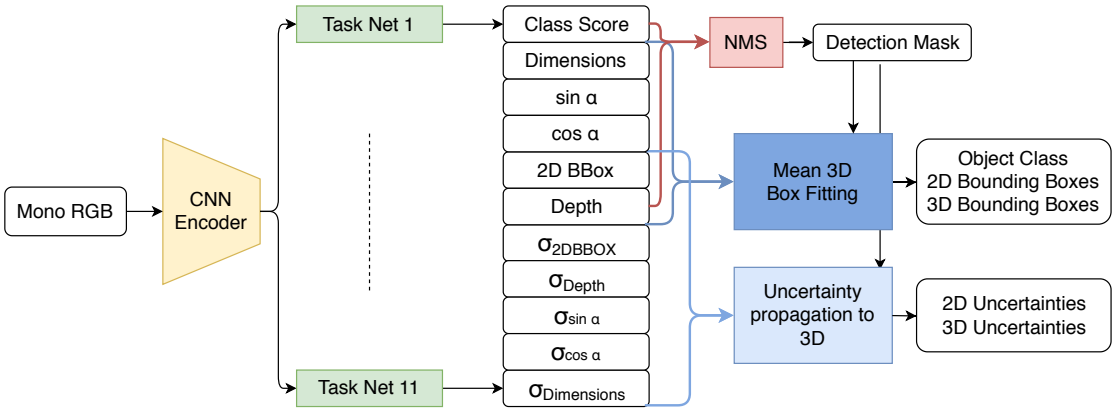


Figure 3.3: The overall network structure.

uncertainties for all three of the world coordinates X, Y and Z whereas now the Z parameter, and corresponding uncertainties, are more isolated.

As described in Section 2.4.2 it is hard to estimate the global orientation, θ, around the Y axis right away and thus we choose to estimate the observation angle, α, to an object. We chose to encode the angle α as both $\sin\alpha$ and $\cos\alpha$. The observation angle is then recreated by

$$\alpha = \arctan\left(\frac{\sin\alpha}{\cos\alpha}\right) \tag{3.4}$$

where $\sin\alpha$ and $\cos\alpha$ are the two network orientation estimates. The final global orientation is then calculated as per Equation (2.16) from Section 2.4.5.

The dimension outputs are estimations of the object dimension in meters straight away. Some papers choose to encode the dimension variables and instead estimate the logarithm of the dimensions to make sure not to get negative dimension estimates. This is not done in this thesis as the outputs are assumed to be affected by observation noise and thus it is favorable to not have to convert the uncertainty estimates from uncertainty in log dimensions to uncertainty in dimensions. We did not experience any problems with the network outputting negative dimensions.

We assume that there is observation noise on the outputs, which is modelled by the aleatoric uncertainties, corrupting each one of the regression parameters (2DBBOX, depth, orientation, dimensions). This observation noise is estimated by the network. The aleatoric uncertainties are parameterized as the logarithm of the standard deviation, $\log\sigma$, instead of estimating the standard deviation σ right away, thus forcing the σ estimates to be positive. Positive uncertainty estimates are desirable as standard deviations and variances for distributions can't be negative.

3.1.3 Deriving 3D variables

Some of the parameters describing a 3D bounding box is not estimated from the network and thus have to be derived from the variables that are estimated. This is done entirely in accordance with Section 2.4.5 resulting in the final parameters describing the 3D bounding box, namely its dimensions (H, W, L), location (X,Y,Z),

rotation (θ). The step of calculating the remaining 3D variables is performed after NMS in order to substantially decrease the computational power that is required.

3.1.4 Propagating uncertainties to 3D

When 3D parameters are derived as a post processing step rather than estimated from the network we have no direct estimate of the uncertainties for these variables and thus the uncertainties has to be propagated in a similar fashion to how the 3D variables are derived. In this Section we will go through how we propagate the uncertainties to the final 3D variables, i.e. how we derive σ for *H, W, L, X, Y, Z* and θ.

The object's dimensions H, W, L are estimated from the network and thus the estimated uncertainties for these variables does not have to be propagated. As described in Section 2.4.5 the 3D location is split between Z which is estimated directly and X and Y which are derived from the 2D bounding box estimates. Similarly to the dimension estimates, the Z estimate is already in the global world coordinate system and thus no propagation of uncertainties is needed.

The variables X and Y are derived from the center point of the 2D bounding box which in turn is derived from the 2D bounding box as in Equation 2.13. The uncertainties about the four 2D bounding box edges are uncertainties in pixels in the image plane. We want to propagate this uncertainty to uncertainty for the world coordinates X and Y, that both have unit meter. To find the uncertainties in X the 2D bounding box center, c_{center}, is shifted in the x direction by $-\sigma_{x_{\min}}$ and $+\sigma_{x_{\max}}$. The world coordinates that corresponds to these two points are found by applying Equation (2.14) followed by Equation (2.15). As the pixel coordinates were two standard deviations apart in the x direction the two world coordinates should now also be two standard deviations apart in the X direction. The standard deviation for the Y world coordinate is found in a similar fashion but shifting c_{center} in the y direction by $-\sigma_{y_{\min}}$ and $+\sigma_{y_{\max}}$ instead of the shift in the x direction.

Finally the uncertainties for the yaw angle θ has to be derived. The network outputs connected to the observation angle α are encoded as $\sin(\alpha)$ and $\cos(\alpha)$. As the estimated uncertainties are also uncertainties in $\sin(\alpha)$ and $\cos(\alpha)$ we have to first convert these to uncertainties in α. This is done by projecting the uncertainties in $\sin(\alpha)$ and $\cos(\alpha)$ onto the unit circle (see Figure 3.4) and taking the absolute value to account for negative values of $\sin(\alpha)$ and $\cos(\alpha)$. These are then considered as separate sources of uncertainty in α, i.e. their variances are added to obtain the variance in α. Note that the uncertainties $\sigma_{\alpha,sin}$ and $\sigma_{\alpha,cos}$ are assumed to be small. The smaller the uncertainties are, the better the approximation is.

3.2 Implementation

The implementation is written in Python and more specifically using PyTorch [41] to build and train the models. The data set that the algorithms have been be trained and evaluated on is the KITTI [33] data set.

The learning rate used during training is initialized to 10^{-4} and a learning rate scheduler is used which reduces the learning rate by a factor 10 when either the F1

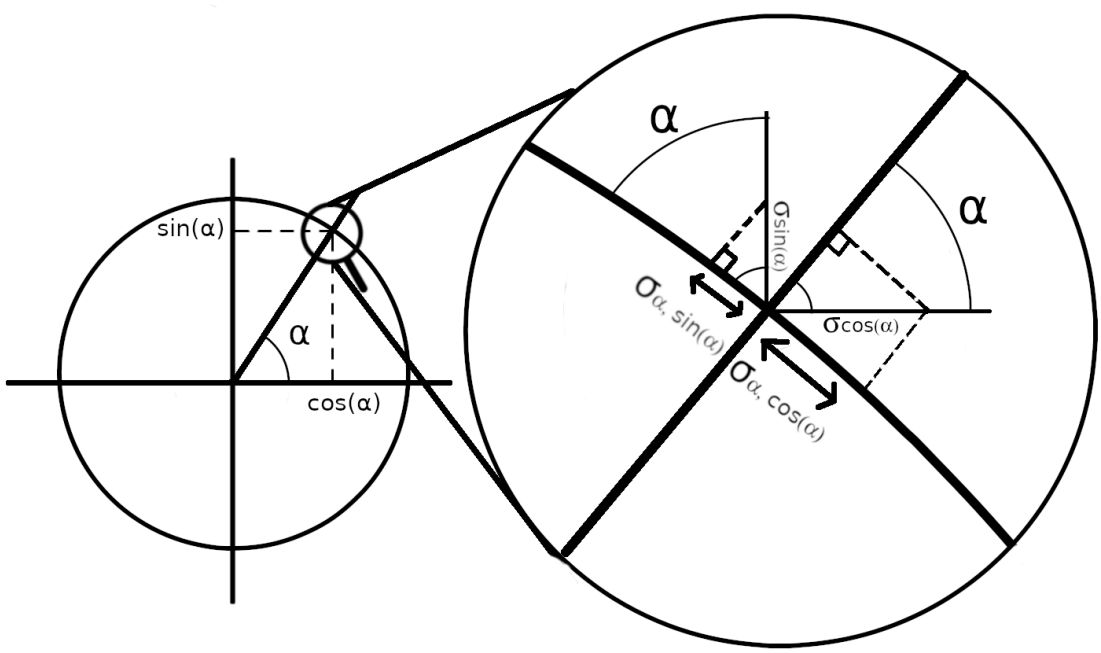


Figure 3.4: Illustration of how uncertainties are propagated to α from sin α and cos α under the assumption that the uncertainties in sin α, cos α are small.

score or the validation loss has not improved for 10 epochs.

Monte Carlo (MC) Dropout is used for approximating the posterior distribution over the weights in the network, as described in Section 2.1, and the dropout rate used during both training and inference is set to 0.2, as suggested in the paper by Kendall and Gal [1]. The number of MC Dropout samples is set to 20.

For non maximum suppression (NMS) a 2D IoU threshold of 0.3 is used when determining if detections belong to the same object.

3.3 Evaluation

Different metrics are used to evaluate and analyze the estimates. The metrics are divided into uncertainty and performance metrics.

3.3.1 Uncertainties

There is currently no standardized way of evaluating uncertainty estimates in machine learning and therefore no comprehensive comparison to other methods could be done. Instead we have to find metrics that covers, and describes, different aspects of the uncertainties to be able to analyze them in a good way. For evaluation and analysis of the estimated uncertainties, Area Under Sparcification Error (AUSE) (see Section 2.6.1), Calibration plots (see Section 2.6.2) and Uncertainty Error (UE) (see Section 2.6.3) are used. Together they cover evaluation of the order of uncertainties in relation to each other, the magnitude (or calibration) of the uncertainties as well as their ability to distinguish true from false positives. The uncertainties will be

evaluated for epistemic and aleatoric uncertainties separately as well as combined.

For AUSE and calibration plots only true positives are considered as they require ground truth targets to be calculated. To be able to evaluate and analyze the uncertainties for as many detections as possible the 2D IoU scoring function is used to distinguish true from false positives as this accepts more detections compared to 3D or BEV scoring functions. Similarly, no more extensive filtering of objects are made than for objects of the class *Car*. In other words, no KITTI benchmark filters, described in Section 2.5.1, are used.

A model is trained under the assumption that there is either Laplacian or Gaussian observation noise corrupting the outputs. To analyze the calibration of a model trained under the assumption that there is Laplacian observation noise corrupting the outputs two set of calibration plots are created. The first set of calibration plots is created by treating each output as a Laplacian distribution characterized by the estimated mean and the estimated variance and then using Equation 2.24 as described in Section 2.6.2. The second set of calibration plots is created by instead treating each output as a Gaussian distribution characterized by the same estimated mean and the same estimated variance, thus using Equation 2.24 as described in Section 2.6.2. These two sets of calibration plots are created for each of the different models. Furthermore, the Mean Squared Error to perfect calibration ($y = x$) is calculated for each variable to be able to compare the calibration across models and parameters.

For UE the ability of different parameters to distinguish true from false positives are evaluated. Since the scoring function alone is what determines if a detection is a true positive or a false positives the UE evaluation is performed with multiple different scoring functions. No KITTI benchmark filters are used but the evaluation is only done for cars as the vast majority of the objects in the dataset are cars.

3.3.2 Performance

For performance the metric of Average Precision (AP) (see Section 2.5.5) and Detection Statistics (see Section 2.5.3) are used. For both metrics all KITTI benchmark filters (see Section 2.5.1) are considered as well as all scorers (see Section 2.5.2). For Average precision, common practise is to use scoring threshold 0.5 but for the KITTI benchmark 0.7 is specified. To possibly gain additional insights as to how close the detections are to becoming true positives the threshold value 0.3 was used as well, this gives some indication of how big the uncertainties have to be in order to cover the otherwise missed objects. In other words, the threshold values 0.3, 0.5, and 0.7 are considered. For Detection Statistics the classifications presented are true positives, false positives and false negatives.

4
Results

In this chapter the results from our different models and experiments will be presented. The baseline for which we compare the performance against is the base model that has been extended with uncertainty estimates, but without uncertainties. Furthermore, the model that we have extended to estimate epistemic and heteroscedastic aleatoric uncertainties is here called UncertaintyNet (UN). A few different versions of UN have been trained with slight differences, these are:

- **UN Val Laplace** - UncertaintyNet with assumed Laplacian observation noise that is stopped on lowest validation loss
- **UN F1 Laplace** - UncertaintyNet with assumed Laplacian observation noise that is stopped on best F1 score (for validation set)
- **UN Val Gauss** - UncertaintyNet with assumed Gaussian observation noise that is stopped on lowest validation loss
- **UN F1 Gauss** - UncertaintyNet with assumed Gaussian observation noise that is stopped on best F1 score (for validation set)

where stopped on means that that model has been chosen as the "best model" for which the presented results in this section are generated from. Everything in this chapter is from experiments on the validation dataset.

4.1 Visualizations / Images

In the following section an example image generated by the UncertaintyNet model trained under Laplacian observation noise that is stopped on best F1 score will be presented. The same image will be used multiple times from different views and with different estimates visualized.

In Figure 4.1 an example image is visualized with 2D ground truth annotations, depth estimates $\pm\sigma$, mean 2D bounding box estimates and two extra boxes per detection where the mean box has been shifted $\pm\sigma$ for each edge.

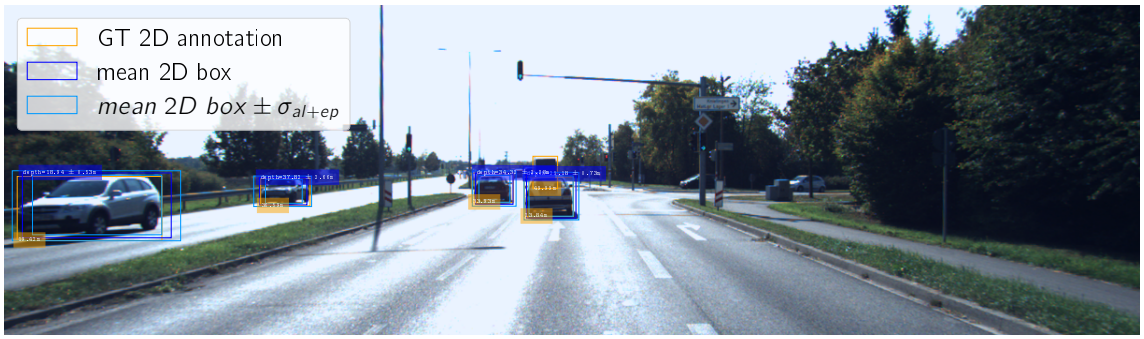


Figure 4.1: An example image with corresponding GT 2D bounding boxes (orange), mean estimated 2D bounding boxes (dark blue) and two additional boxes where the mean boxes has been shifted by one standard deviation (light blue). GT depth is written in the bottom left corner and estimated depth ± one standard deviation is written in top left corner. The standard deviations is here combined epistemic and aleatoric uncertainties for the parameters.

From Figure 4.2 we see that the network indicates greater uncertainty about its depth estimate for the car that is the furthest away and that the uncertainties for the 2D bounding box predictions are greater for the closest car. The estimated 3D bounding boxes are increased in size by one standard deviation and shifted one standard deviation along the X, Y and Z axes to illustrate what volume that is covered within one standard deviation. This volume, along with the estimated 3D bounding box and the ground truth 3D box, is projected onto the image and visualized in Figure 4.3.

From Figure 4.2 we see that the estimated standard deviation for the depth is 2.66m for the car furthest away compared to 0.53m for the closest car. The depth is again the distance along the Z axis. The effect that this has in 3D can clearly be seen in Figure 4.4 where the light blue box for the car with high depth uncertainty is substantially longer than that of the car with small depth uncertainty. The same image is visualized from a bird's-eye-view (BEV) to be able to more easily see how well the estimated 3D boxes match the GT and to see how the uncertainties in various variables contributes to the area covered within one standard deviation. In Figure 4.5 the uncertainties for (W, L), (X, Z) and θ are visualized separately. Furthermore also the aleatoric, epistemic and combined uncertainties are visualized separately. Note that the estimated boxes are manipulated with one standard deviation for each variable and that the combined uncertainty is the square root of the sum of the individual variances, i.e. $\sigma_{al+ep} = \sqrt{\sigma_{al}^2 + \sigma_{ep}^2}$.

The uncertainties for W, L, X, Z and θ can be combined to visualize the total area within one standard deviation for all of these variables combined. This is done in Figure 4.6 where the estimated bounding boxes are made one standard deviation longer and wider, rotated ± one standard deviation around the Y axis and finally shifted ± one standard deviation in the X and Z direction. A few more typical example images and corresponding bird's-eye-view visualizations for the *UN F1 Laplace* model can be seen in Figures 4.7 - 4.10.

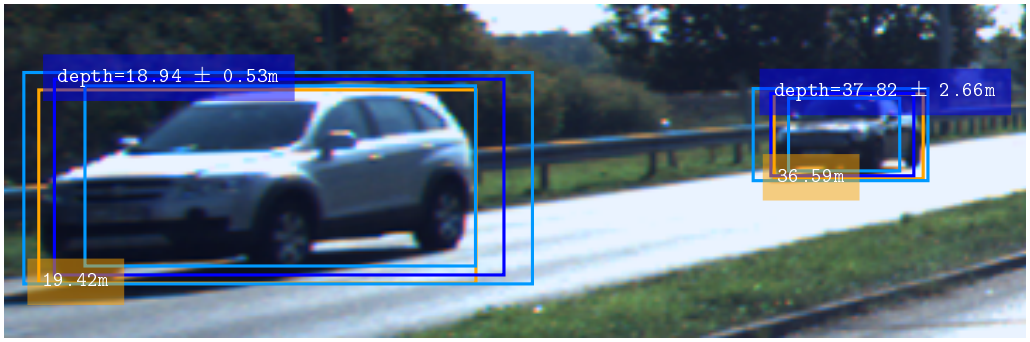


Figure 4.2: Two detections with the GT 2D bounding box (orange), mean estimated 2D bounding box (dark blue) and mean box shifted one standard deviation for each edge (light blue). Ground truth depth is written in the bottom left corner and mean depth ± one standard deviation in the top left corner.
The mean estimates (2D bounding box and depth) for these objects are within one standard deviation of the ground truth annotations, except for the top edge on the closest car that is one pixel below the uncertainty box.

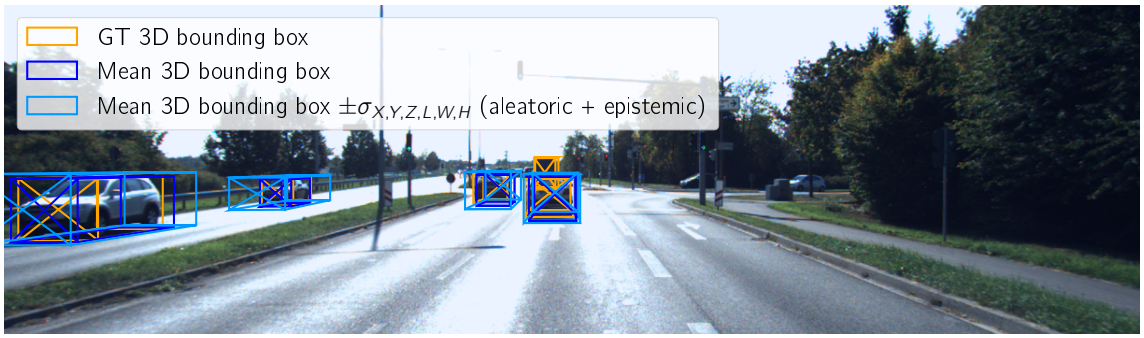


Figure 4.3: An example image with GT 3D bounding boxes (orange), estimated 3D bounding boxes (blue) and the volumes created by increasing the size of each blue box by one standard deviation in all dimensions and then shifting them one standard deviation along the world coordinate axes X,Y,Z (green). σ is here combined epistemic and aleatoric uncertainties for the parameters.

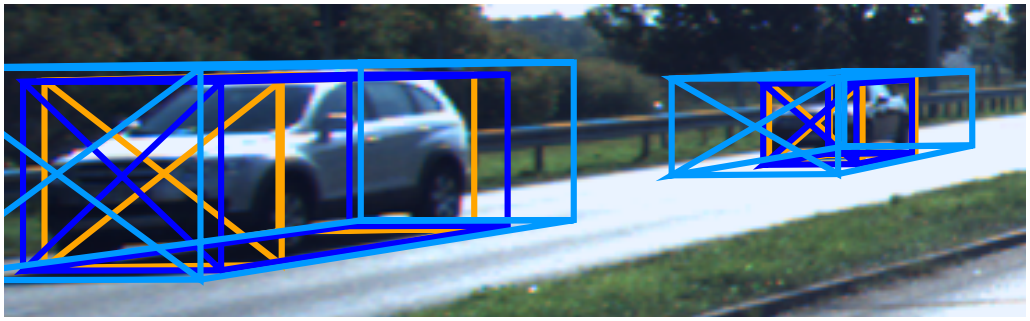


Figure 4.4: Two detections with the GT 3D bounding boxes (orange), estimated 3D bounding boxes (blue) and the volume created by increasing the size of the blue boxes by one standard deviation in all dimensions and then shifting it one standard deviation along the world coordinate axes X,Y,Z (green).

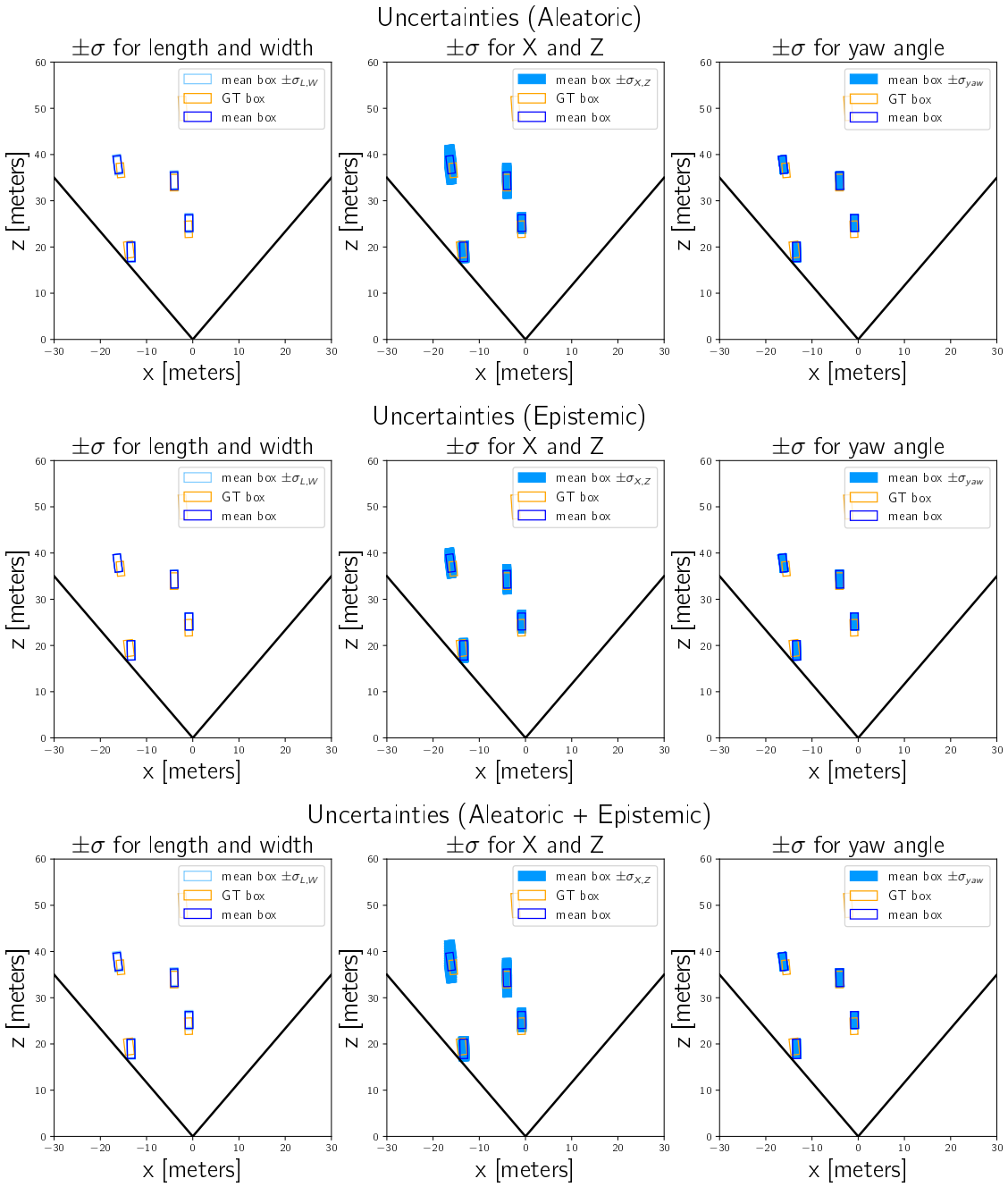


Figure 4.5: A bird's-eye-view visualization of the area covered within one standard deviation for the aleatoric uncertainties (top), epistemic uncertainties (middle) and combined aleatoric and epistemic uncertainties (bottom). The estimated bounding boxes (dark blue) have been manipulated to show the area (light blue) within one standard deviation for three different parameter settings. The three plots show: boxes increased/decreased in width and length with one standard deviation (left), boxes shifted one standard deviation in the X and Z direction (middle) and boxes rotated one standard deviation around the Y axis (right). Note that the uncertainties for the length and width parameters are so small that they are hardly visible.

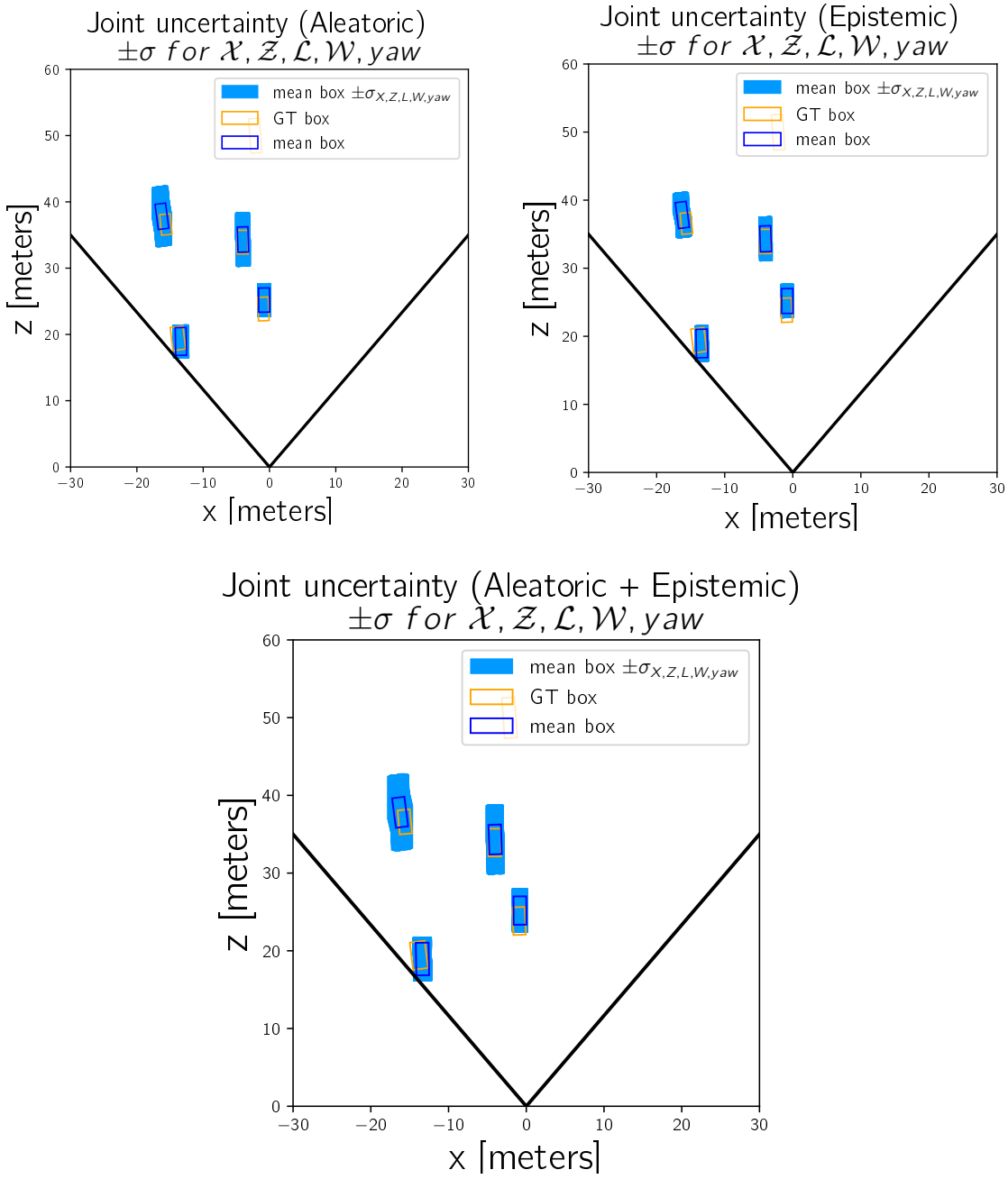


Figure 4.6: A bird's-eye-view visualization of the area covered within one standard deviation for the aleatoric uncertainties (top left), epistemic uncertainties (top right) and combined aleatoric and epistemic uncertainties (bottom). The estimated bounding boxes (dark blue) have been manipulated to show the area (light blue) within one standard deviation for W, L, X, Z and θ. The boxes are made one standard deviation longer and wider, rotated ± one standard deviation around the Y axis and then shifted ± one standard deviation in the X and Z direction.

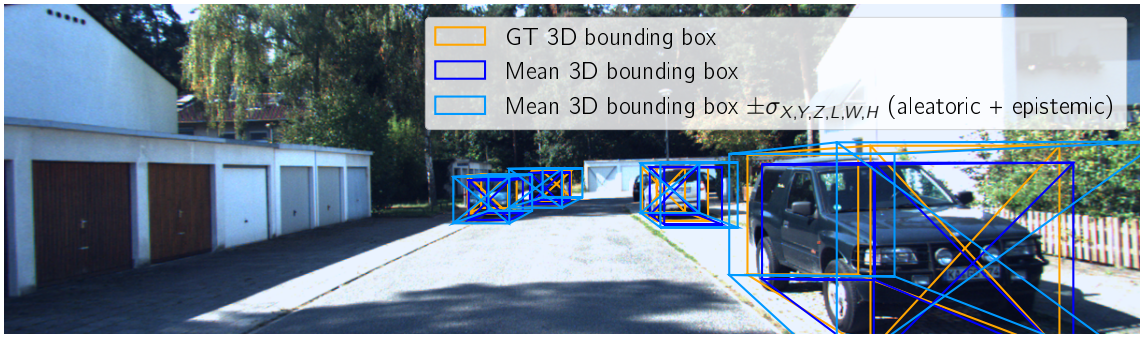


Figure 4.7: An example image with GT 3D bounding boxes (orange), estimated 3D bounding box (blue) and the volume created by increasing the size of the blue box by one standard deviation in all dimensions and then shifting it one standard deviation along the world coordinate axes X,Y,Z (green). σ is here combined epistemic and aleatoric uncertainties for the parameters. The corresponding bird's-eye-view version of this image can be seen in Figure 4.8.

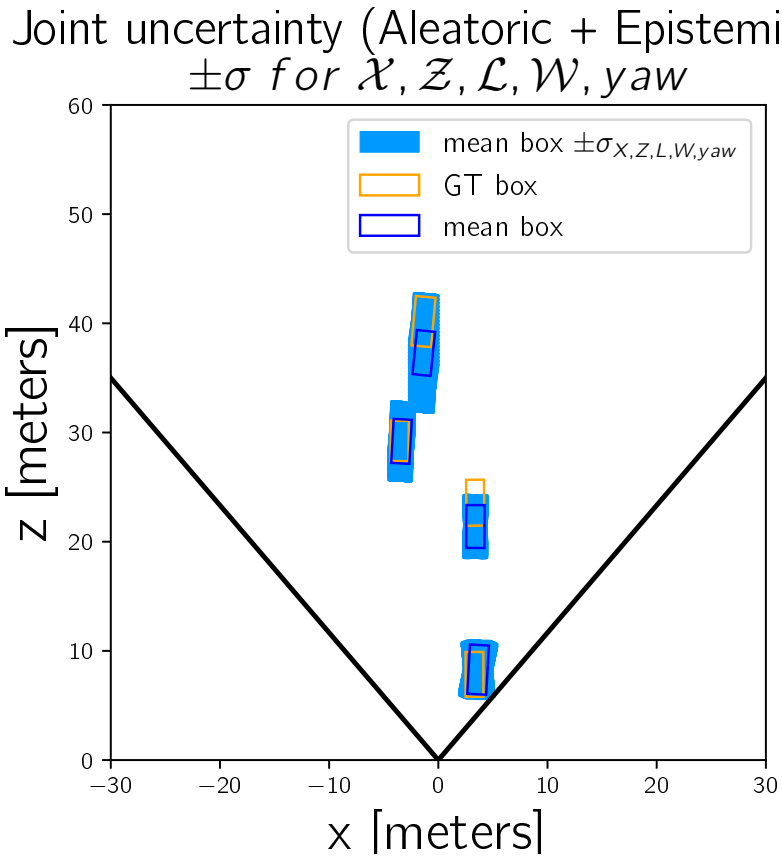


Figure 4.8: A bird's-eye-view version of Figure 4.7 to visualize the area covered within one standard deviation for the combined aleatoric and epistemic uncertainties. The estimated bounding boxes (dark blue) have been manipulated to show the area (light blue) within one standard deviation for W, L, X, Z and θ. The boxes are made one standard deviation longer and wider, rotated $\pm$ one standard deviation around the Y axis and then shifted $\pm$ one standard deviation in the X and Z direction.

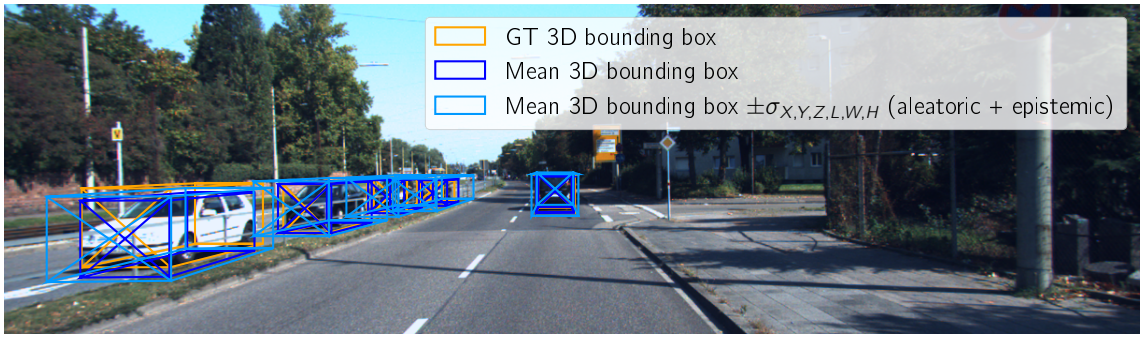


Figure 4.9: An example image with GT 3D bounding boxes (orange), estimated 3D bounding box (blue) and the volume created by increasing the size of the blue box by one standard deviation in all dimensions and then shifting it one standard deviation along the world coordinate axes X,Y,Z (green). σ is here combined epistemic and aleatoric uncertainties for the parameters. The corresponding bird's-eye-view version of this image can be seen in Figure 4.10.

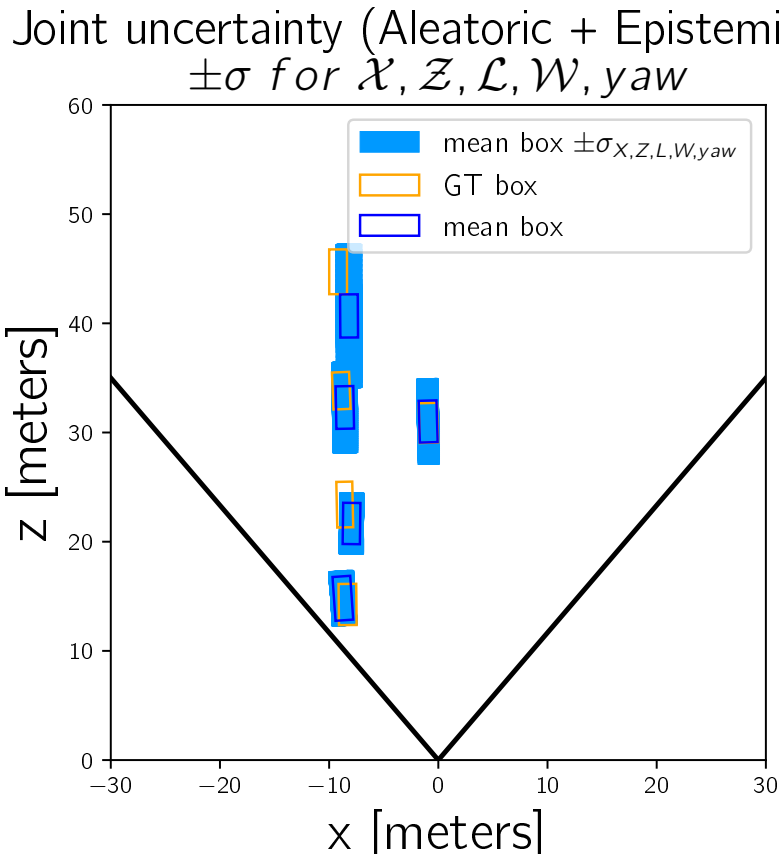


Figure 4.10: A bird's-eye-view version of Figure 4.9 to visualize the area covered within one standard deviation for the combined aleatoric and epistemic uncertainties. The estimated bounding boxes (dark blue) have been manipulated to show the area (light blue) within one standard deviation for W, L, X, Z and θ. The boxes are made one standard deviation longer and wider, rotated ± one standard deviation around the Y axis and then shifted ± one standard deviation in the X and Z direction.

4.2 Uncertainty metrics

For evaluating uncertainties different metrics cover different aspects and complement each other. By combining them a complete analysis of the uncertainties can be made. The metrics used are *Area Under Sparcification Error*, *Calibration* and *Uncertainty Error*. Throughout this section the same model as for the visualizations in Section 4.1 will be used, more precisely the *UN F1 Laplace model*.

4.2.1 Area Under Sparcification Error

A few sparcification curves demonstrating different phenomena are presented in Figures 4.11, 4.12 and 4.13. What can be seen in Figure 4.11 is that the sparcification curves for the aleatoric and the epistemic uncertainty in the rotation parameter differs drastically for the most certain estimates. The length parameters (Figure 4.12) has the highest AUSE for this model and, out of the 3D parameters, the Z parameter has the lowest AUSE for this model (Figure 4.13). Additional sparcification curves can be found in Appendix A.

The three Area Under Sparcification Error curves for aleatoric, epistemic and combined uncertainties can be seen in Figure 4.14. What can be observed is that length, width and Y have the highest and Z, 2DBBOX xmin and 2DBBOX xmax the lowest values for AUSE.

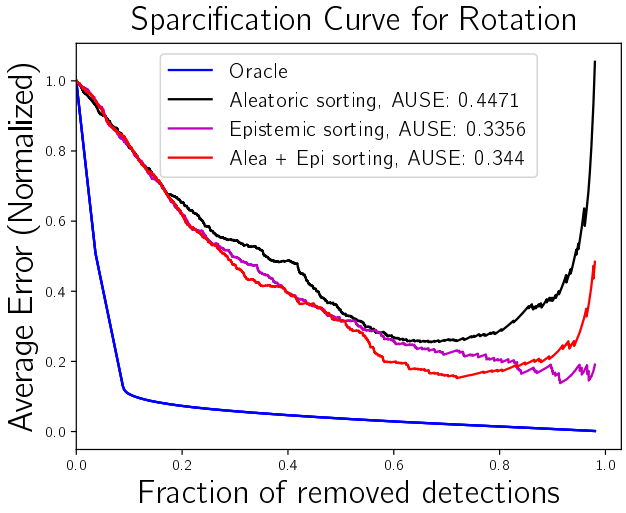


Figure 4.11: Sparcification curves for the rotation parameter.

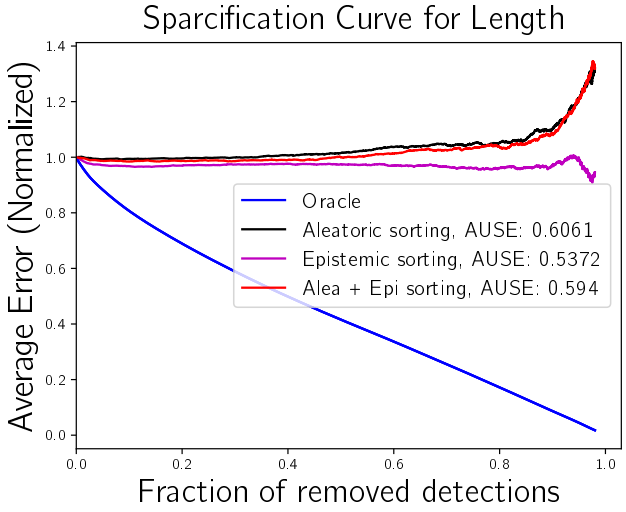


Figure 4.12: Sparcification curves for the length parameter. This parameter has the highest value for AUSE for the *UN F1 Laplace* model.

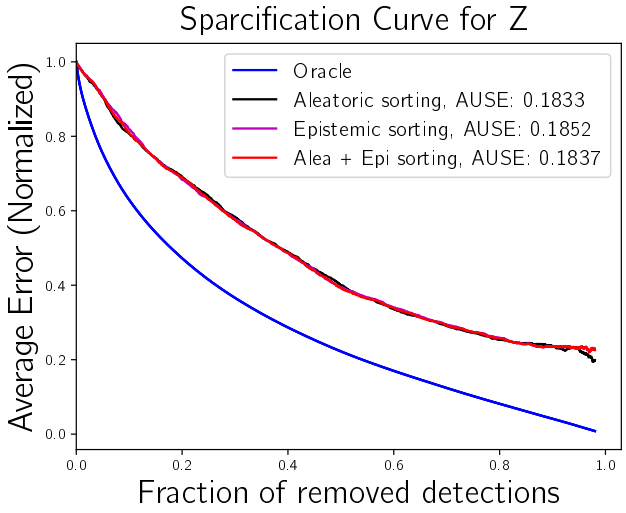


Figure 4.13: Sparcification curves for the Z parameter. This parameter has the lowest value for AUSE out of the 3D parameters for the *UN F1 Laplace* model.

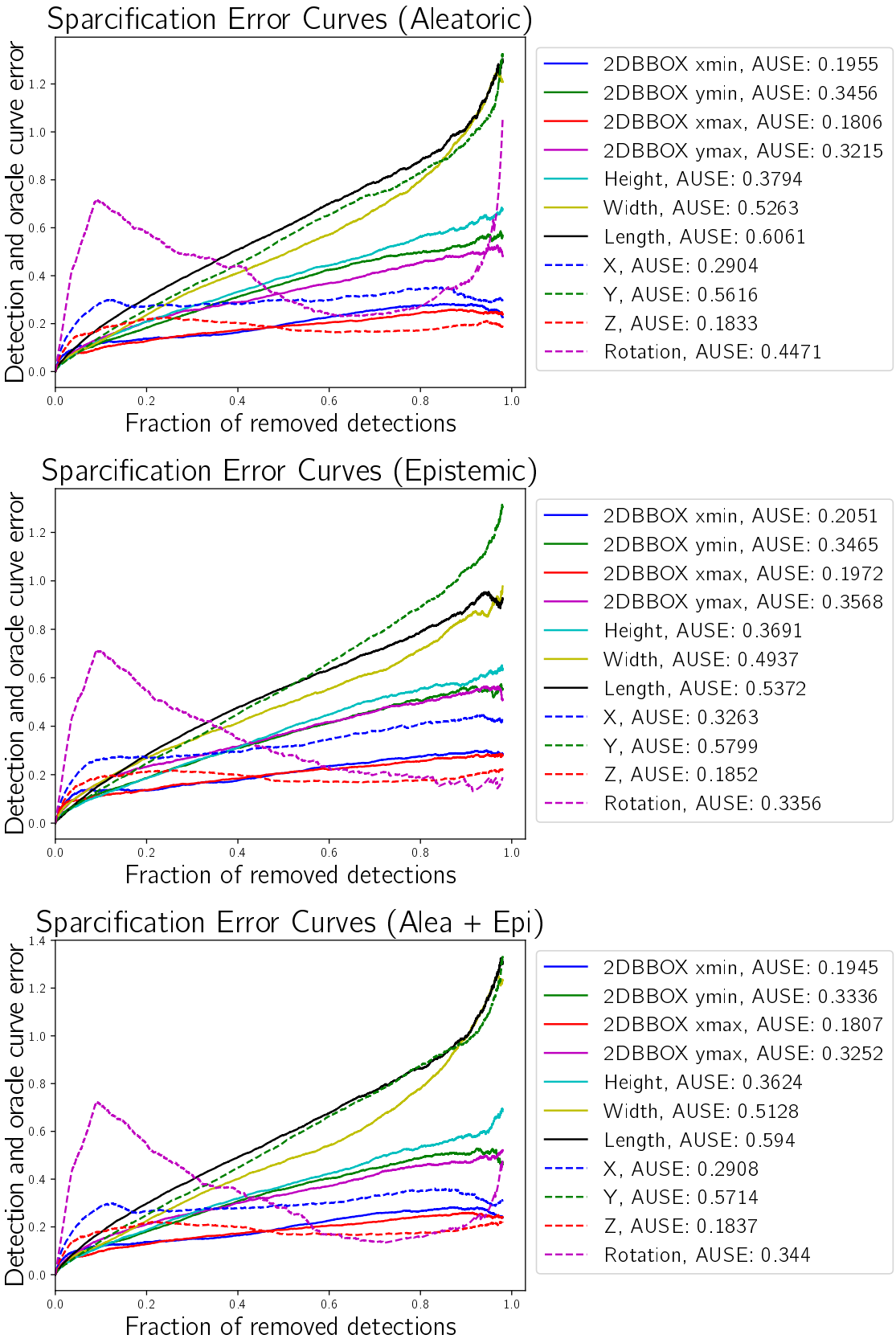


Figure 4.14: Area Under Sparcification Error Curves for all parameters and for aleatoric, epistemic as well as combined uncertainties. The model used is *UN F1 Laplace*.

4.2.2 Calibration

The calibration plots in this section are for the parameters that describe a 3D bounding box, i.e. dimensions, location, rotation, all of which can be seen in Figure 4.15. The model is consistently over-confident which can be seen by that the curves lies below the perfect calibration line $y = x$. In other words, the uncertainty estimates are typically too small, resulting in that the intervals found when integrating the distribution are too narrow and the ground truth does not fall within this narrow interval frequently enough. If the uncertainties would have been bigger the intervals would be wider for the same area under the probability density function and thus the ground truth will fall within the specific intervals more often.

4.2.3 Uncertainty Error

The Laplacian minimum uncertainty error for different IoU scoring functions (2D, 3D and BEV) and uncertainties (Aleatoric, Epistemic and Combined) are presented in Table 4.1. Based on those results the Uncertainty Error plots for the parameters with the lowest (best) MUEs for combined uncertainty for the three different scoring methods are visualized in Figures 4.16 - 4.18. In Figure 4.19 the UE plot for the model and parameter with the highest (worst) MUE is visualized. Additional Uncertainty Error plots can be found in Figures A.8 - A.19 in Appendix A.

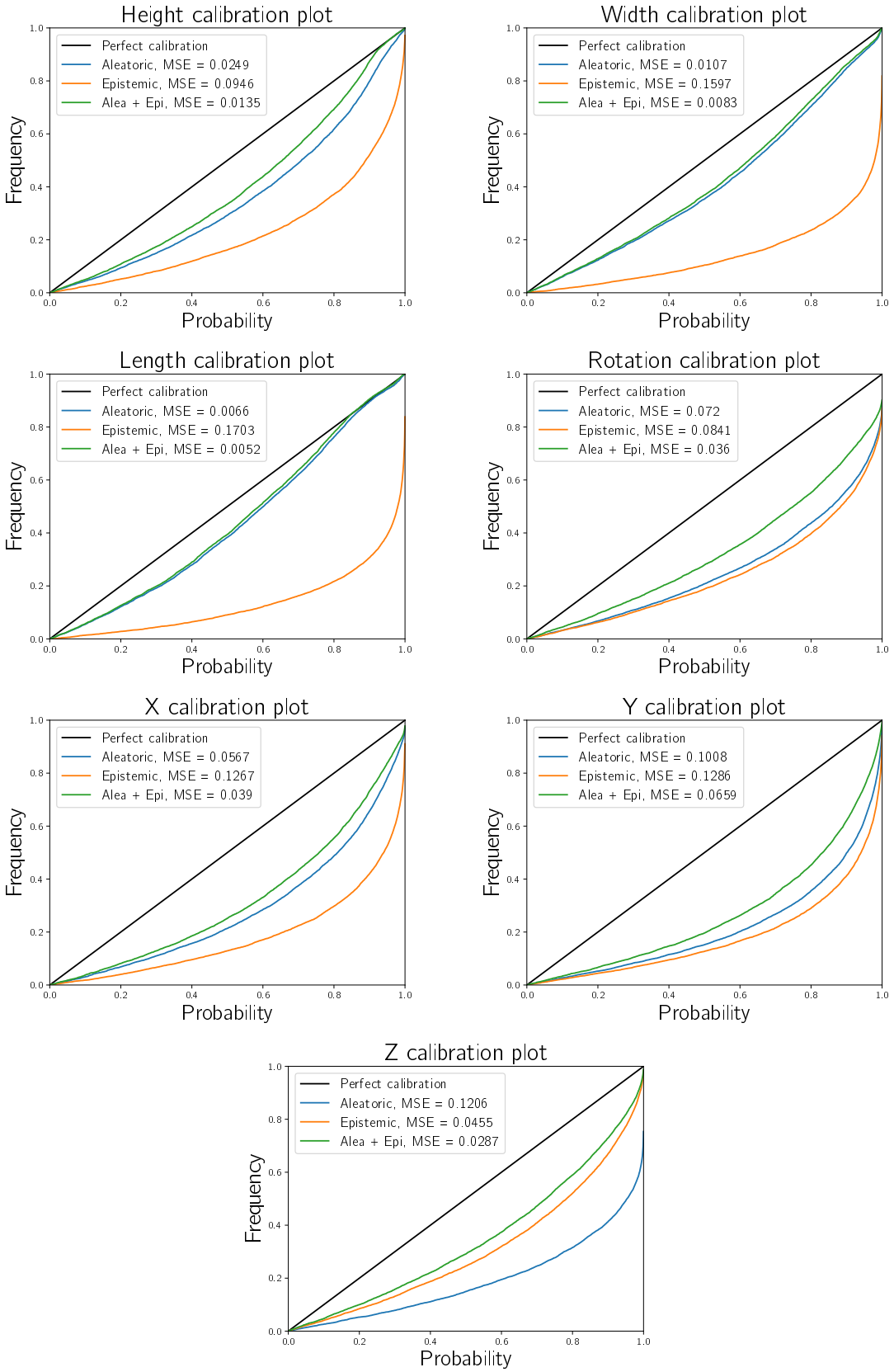


Figure 4.15: Calibration plots of all the 3D variables for the model *UN F1 Laplace*. The model is consistently over-confident seen by that the curves are below the perfect calibration line $y = x$.

Table 4.1: Laplacian Minimum Uncertainty Error (LMUE) for the model UN F1 Laplace when considering different IoU scoring functions (2D, 3D and BEV) and uncertainties (Aleatoric, Epistemic and Combined). The lowest (best) LMUE for a specific scorer and uncertainty is highlighted in bold, i.e. the LMUE for the parameter(s) that best separates true positives from false positives for a specific scorer and uncertainty.

Laplacian Minimum Uncertainty Error

Scorer	**2D IoU**			**3D IoU**			**BEV IoU**		
Uncertainty	**Alea**	**Epi**	**Comb.**	**Alea**	**Epi**	**Comb.**	**Alea**	**Epi**	**Comb.**
2DBBOX in x	.403	.428	.406	.344	.368	**.346**	.395	.405	.396
2DBBOX in y	.335	.382	.353	**.342**	**.355**	.354	.421	.426	.423
2DBBOX	.380	.382	.385	.351	.365	.358	.411	.414	.409
Height	.293	.332	.317	.446	.450	.457	.386	.420	.392
Length	.402	.347	.349	.489	.474	.488	.457	.472	.459
Width	.343	.338	.331	.406	.459	.414	.375	.458	.383
Dimensions	.266	.336	.286	.437	.462	.448	.392	.445	.398
X	.265	.244	.251	.353	.398	.366	**.354**	**.373**	**.350**
Y	**.215**	.237	.217	.423	.447	.437	.405	.439	.414
Z	.305	.288	.283	.482	.472	.473	.445	.434	.438
Location	.249	**.218**	**.216**	.447	.443	.437	.407	.400	.395
Rotation	.316	.438	.366	.485	.480	.482	.462	.472	.471
Dimensions, Location, Rotation	.238	.262	.218	.415	.449	.452	.403	.404	.391
Length, Width, X, Z, Rotation	.251	.251	.241	.465	.443	.459	.410	.419	.403
X, Z	.264	.231	.234	.456	.443	.446	.413	.404	.397

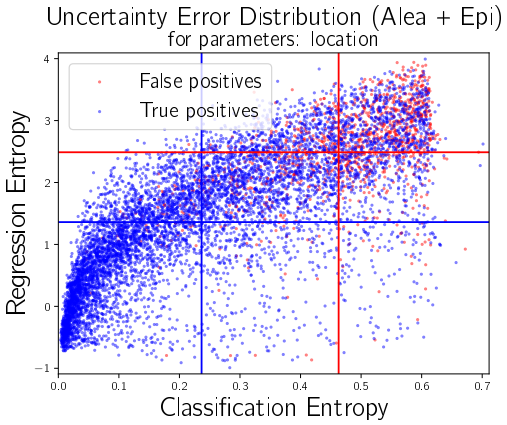


Figure 4.16: Uncertainty Error plot for the *location* parameters with scoring function *2D IoU* and a combination of aleatoric and epistemic uncertainty. This is the parameter with lowest LMUE for the specific scorer.

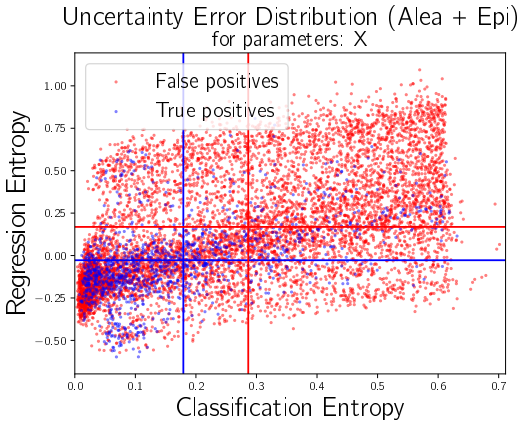


Figure 4.17: Uncertainty Error plot for the X parameter with scoring function *BEV IoU* and a combination of aleatoric and epistemic uncertainty. This is the parameter with lowest (best) LMUE for all scorers.

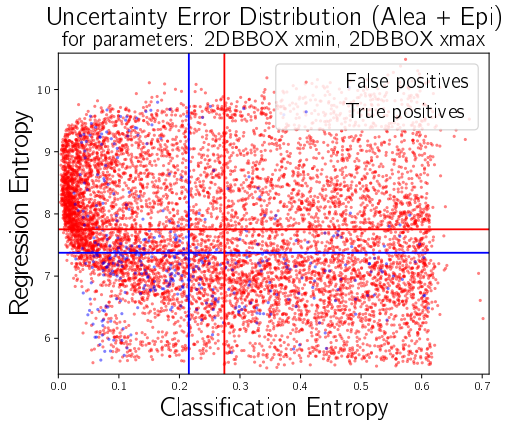


Figure 4.18: Uncertainty Error plot for the *2DBBOX x-edges* with scoring function *3D IoU* and a combination of aleatoric and epistemic uncertainty. This is the parameter with lowest (best) LMUE for the specific scorer.

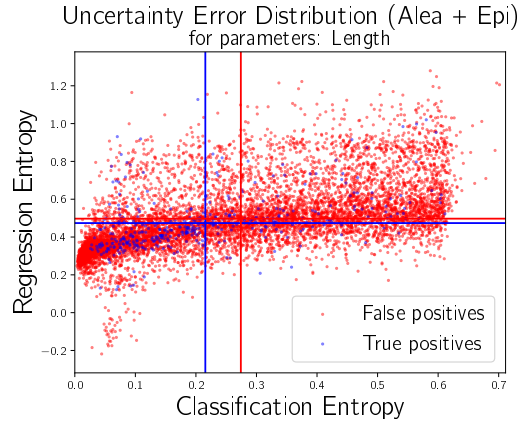


Figure 4.19: Uncertainty Error plot for the *length* parameter with scoring function *3D IoU* and a combination of aleatoric and epistemic uncertainty. This is the parameter with highest (worst) LMUE for all scorers.

4.3 Model comparison

In this section a comparison between different models will be done for the various uncertainty metrics. For the different models, assumptions are made regarding what distribution the outputs belong to and thus what distribution that the uncertainties characterizes. The output noise is either assumed to be Gaussian or Laplacian. Furthermore the models have been stopped on either lowest validation loss or highest F1 score on the validation set, which naturally has an impact on both the performance and the uncertainty estimates.

4.3.1 Area Under Sparcification Error

The sparcification error curves for the combined aleatoric and epistemic uncertainty for the *UN F1 Laplace* model was presented in Figure 4.14. The same plot for the three remaining models (*UN Val Laplace*, *UN F1 Gauss* and *UN Val Gauss*) is presented in Figures 4.20 - 4.22 respectively. What can be seen in these figures is that the AUSE score does not differ substantially between the models except for the AUSE score for the rotation parameter which is exceptionally bad for *UN Val Gauss* and exceptionally good for *UN Val Laplace*.

The minimum, maximum and average AUSE values for each model is presented in Table 4.2 to simplify the process of ranking the models AUSE-wise. What can be seen is that the average error is slightly lower for *UN F1 Laplace* compared to the other models. The lowest AUSE was obtained for rotation by *UN Val Laplace*.

Sparcification curves for some of the extreme parameters for different models can be seen in Figures 4.23 - 4.26. The uncertainty quality, AUSE-wise, for rotation, Z and 2D bounding box xmax is good (low AUSE) as can be seen in the Figures 4.23 - 4.25. However, the rotation parameter for *UN Val Gauss* has a very high AUSE, as can be seen in Figure 4.26.

Table 4.2: Table containing Area Under Sparcification Error for different models. The lowest values are highlighted in bold.

Area Under Sparcification Error

Parameter set	**Metric \Model**	UN F1 Lapl.	UN Val Lapl.	UN F1 Gauss	UN Val Gauss
All	**Min**	0.1806	**0.1444**	0.2064	0.2623
	Max	**0.6061**	0.6221	0.6150	0.9474
	Average	0.3595	**0.3550**	0.3626	0.4455
3D	**Min**	0.1833	**0.1444**	0.2074	0,2623
	Max	**0.6061**	0.6221	0.6150	0.9474
	Average	0.4133	**0.3958**	0.4180	0.5219

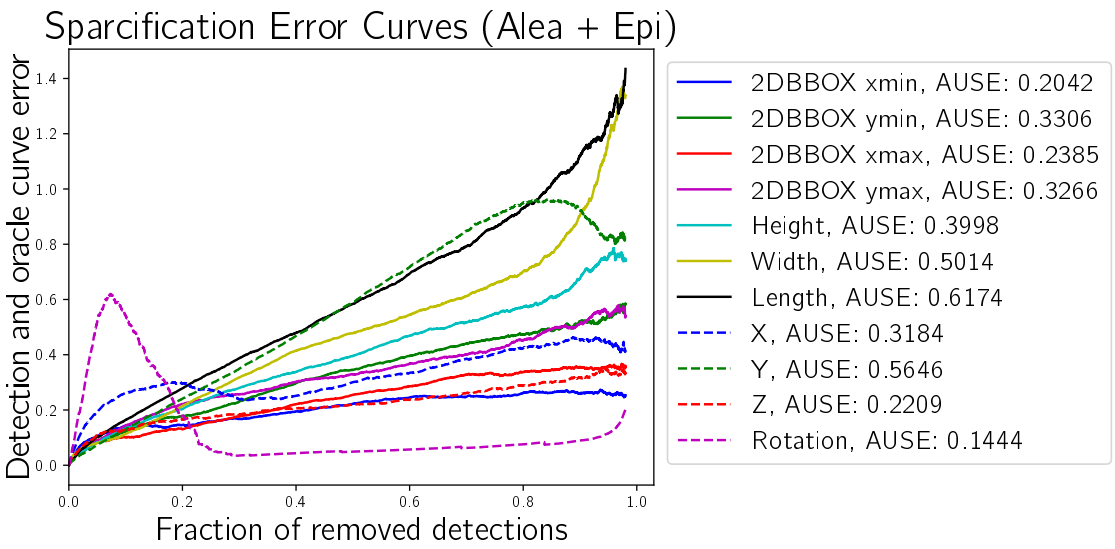


Figure 4.20: Sparcification Error curves for *UN Val Laplace*.

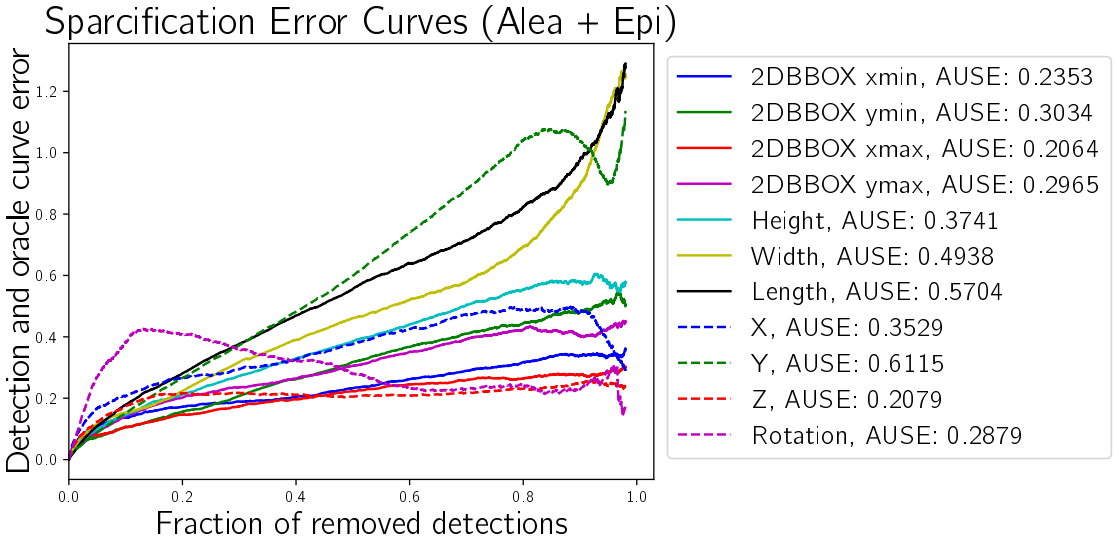


Figure 4.21: Sparcification Error curves for *UN F1 Gauss*.

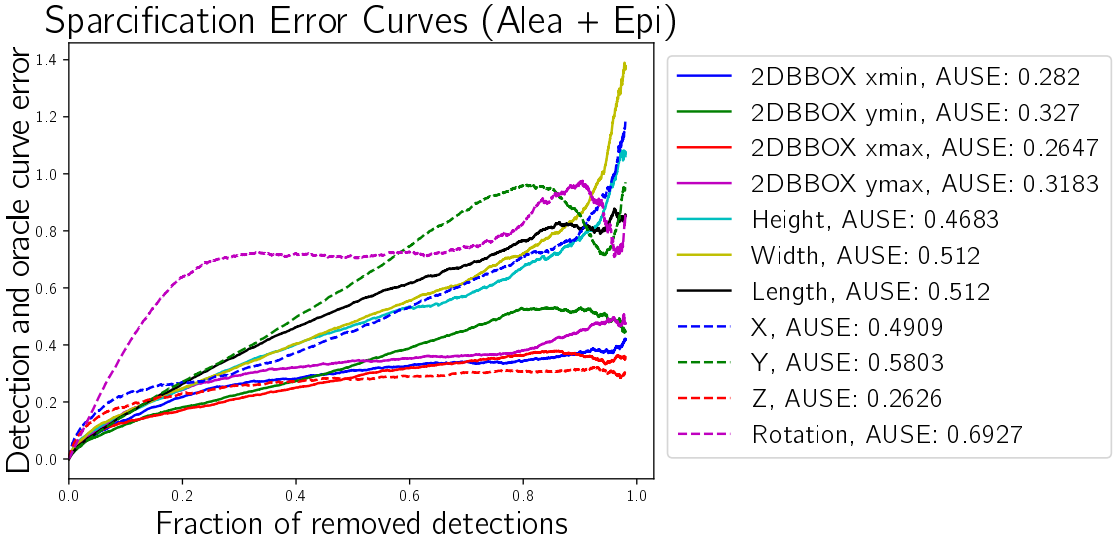


Figure 4.22: Sparcification Error curves for *UN Val Gauss*.

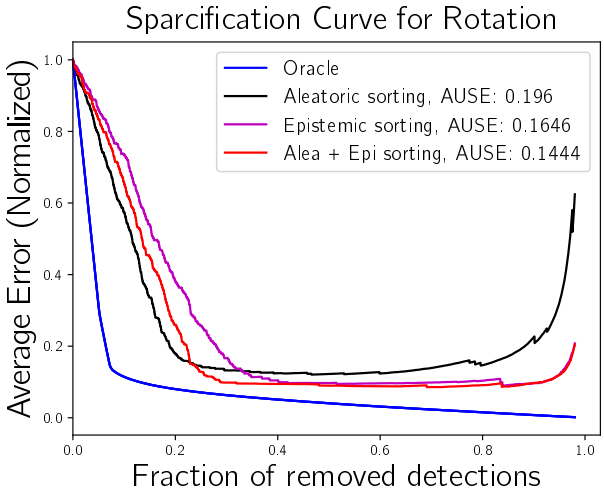


Figure 4.23: Sparcification curves for the rotation parameter for the model *UN Val Laplace*. This parameter has, by far, the lowest value for AUSE for the combined uncertainty.

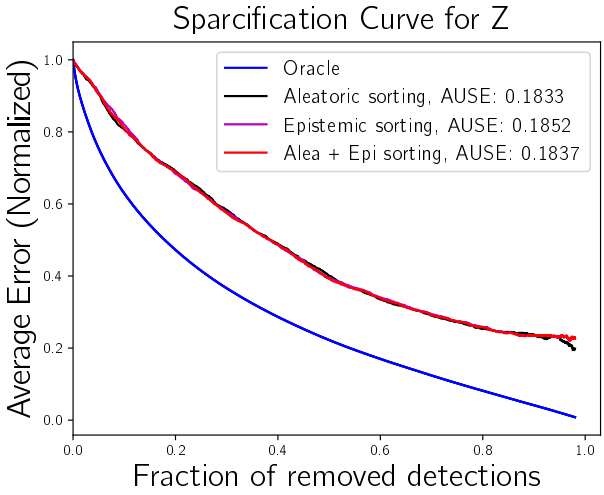


Figure 4.24: Sparcification curves for the Z parameter for the model *UN F1 Laplace*. This parameter has the second lowest value for AUSE.

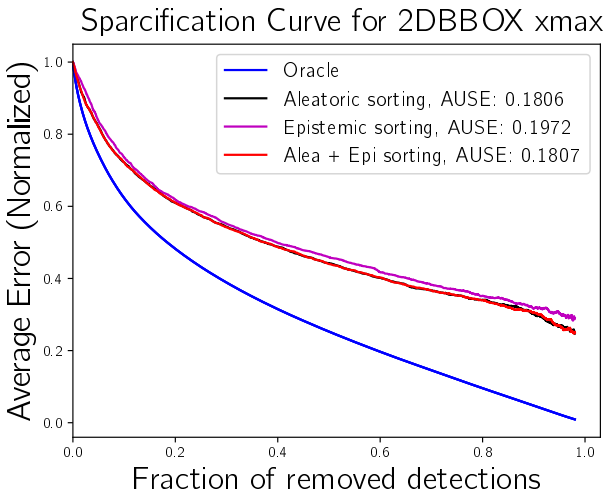


Figure 4.25: Sparcification curves for the 2D bounding box xmax (right edge) parameter for the model *UN F1 Laplace*. This parameter has the lowest AUSE value for the 2D bounding box parameters.

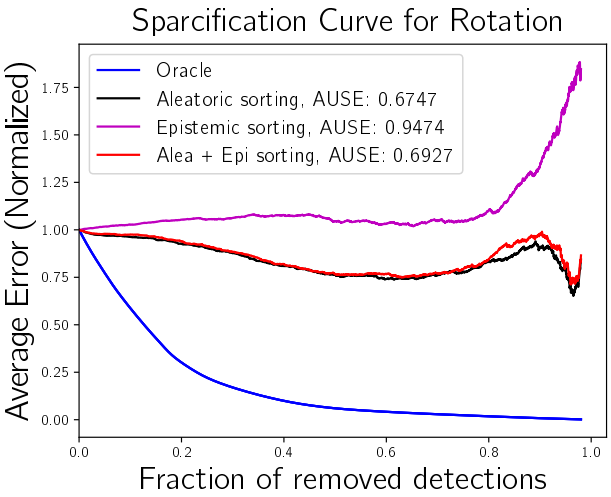


Figure 4.26: Sparcification curves for the rotation parameter for the model *UN Val Gauss*. This parameter has, by far, the highest value for AUSE.

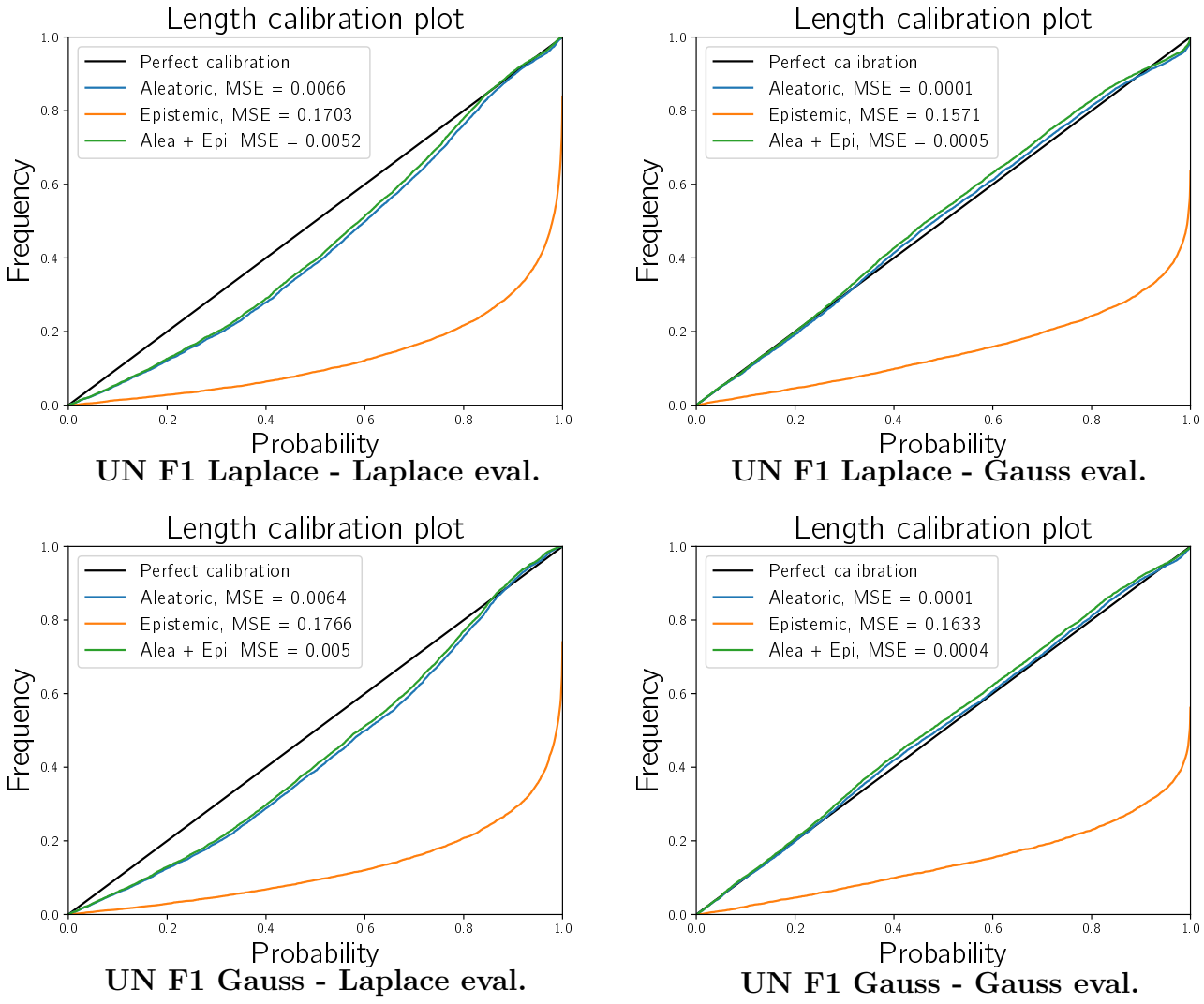
4.3.2 Calibration

Calibration plots for the length parameter is compared in Figure 4.27 where two different models were both stopped on best F1 score. One model is assumed having Laplacian, and the other Gaussian, observation noise, thus being trained with L1 and L2 loss respectively. Both models were, furthermore, both evaluated as a Gaussian and as a Laplacian distribution, thus using Equation (2.23) and (2.24). This is done with the ambition to analyze what distribution the output follows better. The length parameter is rather well calibrated when evaluated as Laplacian but the models evaluated as Gaussian results in an exceptionally well calibrated estimate for the length parameter.

The dimension for cars has a relatively small variance (cars are typically the same size) and the estimates are rather well calibrated. A variable where the variance is much higher is the depth to the object along the Z axis. This estimate is however not as well calibrated in general. A comparison of the calibration for the Z parameter for the four models (Gaussian/Laplace stopped on F1/Val) evaluated as their respective assumed density is done in Figure 4.28. The models that have been stopped on best F1 score (left) are overconfident (curves below perfect calibration). The models that have been stopped on lowest validation loss (right) are under-confident for the aleatoric uncertainties and the combined uncertainties whereas the epistemic uncertainty alone is quite well calibrated.

The MSE for each parameter, each type of uncertainty (aleatoric, epistemic and combined) and for all models evaluated as both Laplacian and Gaussian is presented in Table 4.3. The lowest MSE for each parameter and type of uncertainty is highlighted in bold. Furthermore also the average MSE across the parameters for each model and type of uncertainty is presented, with the lowest average MSE for each type of uncertainty highlighted in bold. The model *UN F1 Gauss* evaluated as a Gaussian has the lowest average MSE for the aleatoric and epistemic uncertainties separately but not for the combined uncertainty where the same model but evaluated as a Laplacian has a lower average MSE.

In Table 4.4 the average MSE for all the 3D parameters (H, L, W, X, Y, Z, θ) for the different models evaluated as either a Laplacian or as a Gaussian distribution is presented. The calibration plots for these parameters for *UN F1 Laplace* can be seen in Figure 4.15 and for the remaining models (*UN Val Laplace*, *UN F1 Gauss* and *UN Val Gauss*) in Figures A.3, A.6 and A.7 in Appendix A.



UN F1 Laplace - Laplace eval. **UN F1 Laplace - Gauss eval.**

UN F1 Gauss - Laplace eval. **UN F1 Gauss - Gauss eval.**

Figure 4.27: Calibration plots for the length parameter for two different models, that were stopped on best F1 score, evaluated as two different distributions; *UN F1 Laplace* evaluated as Laplace (top left); *UN F1 Laplace* evaluated as Gaussian (top right); *UN F1 Gauss* evaluated as Laplace (bottom left); *UN F1 Gauss* evaluated as Gauss (bottom right). The length estimates are better calibrated when the models are evaluated as if the length has a Gaussian distribution compared to a Laplacian distribution.

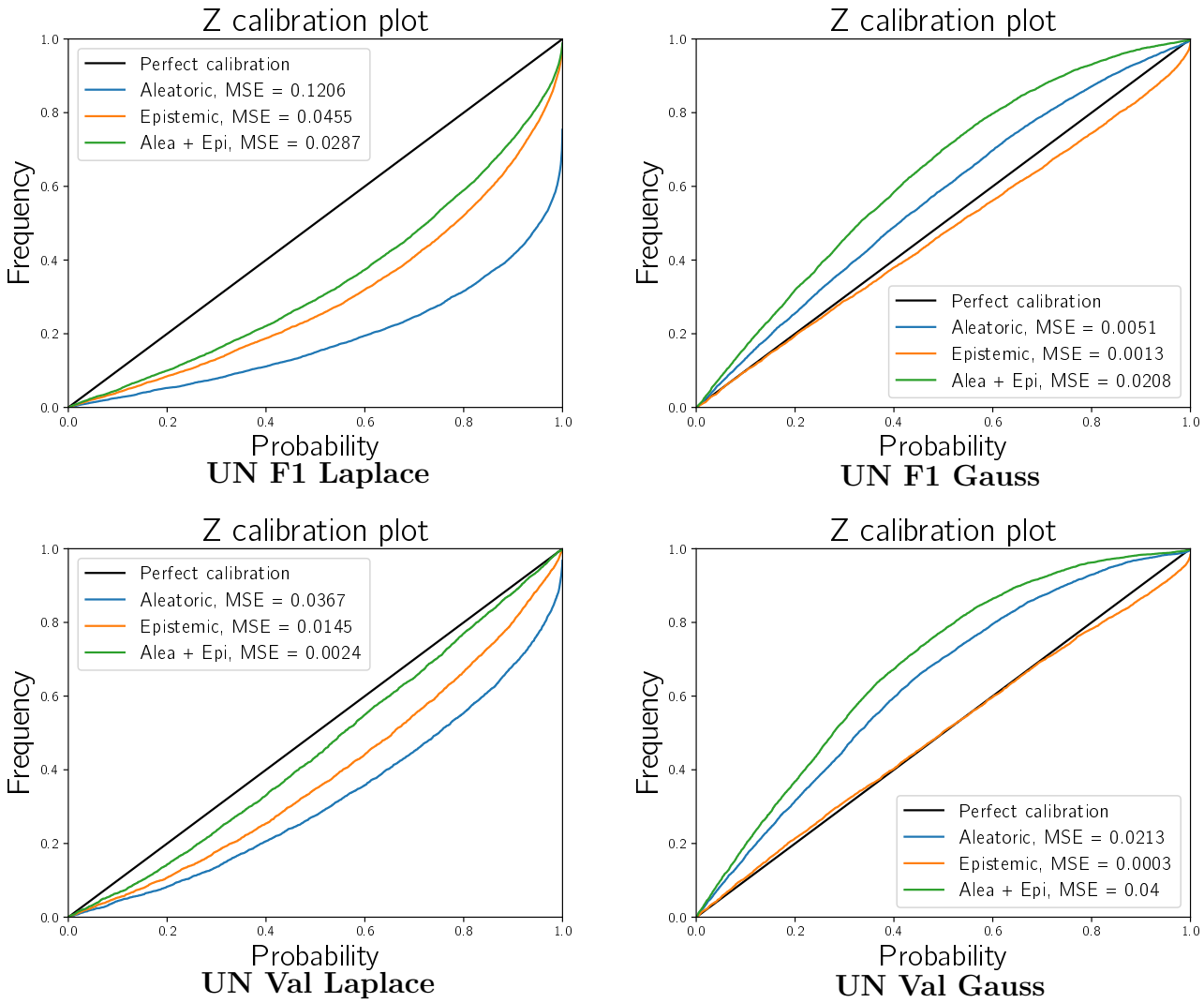


Figure 4.28: Calibration plots for the Z world coordinate for four different models either trained with assumed Laplacian or Gaussian observation noise and that were stopped on two different criteria, either on lowest validation loss or best F1 score. All four models are evaluated as their assumed density; *UN F1 Laplace* (top left); *UN F1 Gauss* (top right); *UN Val Laplace* (bottom left); *UN Val Gauss* (bottom right). The length estimates are better calibrated when the models are evaluated as if the length has a Gaussian distribution compared to a Laplacian distribution.

Table 4.3: The Mean Squared Error (MSE) to perfect calibration for all parameters, models and types of uncertainties. The lowest MSE for each parameter and type of uncertainty is highlighted in bold. Lapl./Lapl. means that the model was trained as a Laplacian and evaluated as a Laplacian. Similarly Lapl./Gauss means that the model was trained as a Laplacian and evaluated as a Gaussian.

Param	Model / Uncert.	UN F1 Lapl./Lapl.	UN F1 Lapl./Gauss	UN Val Lapl./Lapl.	UN Val Lapl./Gauss	UN F1 Gauss/Gauss	UN F1 Gauss/Lapl.	UN Val Gauss/Gauss	UN Val Gauss/Lapl.
X	**Aleatoric**	.0553	.0351	**.0013**	.0069	.0180	.0356	.0058	.0201
	Epistemic	.1243	.1065	.1410	.1253	.0708	.0906	**.0606**	.0812
	Combined	.0378	.0191	**.0012**	.0086	.0067	.0205	.0013	.0105
Y	**Aleatoric**	.1009	.0806	.0611	.0373	.012	.0298	**.0032**	.0176
	Epistemic	.1290	.1112	.1355	.1192	.0656	.0857	**.0488**	.0695
	Combined	.0659	.0439	.0422	.0196	.0027	.0157	**.00001**	.0077
Z	**Aleatoric**	.1202	.1031	.0367	.0210	.0051	**.0011**	.0213	.0037
	Epistemic	.0451	.0263	.0145	.0032	.0013	.0117	**.0003**	.0071
	Combined	.0285	.0128	.0024	**.0014**	.0208	.0034	.0400	.0118
$\boldsymbol{\theta}$	**Aleatoric**	.0717	.0519	.0119	**.0024**	.0132	.0284	.0421	.0551
	Epistemic	.0834	.0626	**.0427**	.0241	.0536	.0752	.1332	.1497
	Combined	.0354	.0187	.0024	**.0021**	.0047	.0144	.0375	.0488
H	**Aleatoric**	.0240	.0078	.0049	.0008	**.0010**	.0042	.0098	.0035
	Epistemic	.0939	.0736	.1237	.1032	**.0732**	.0915	.1368	.1551
	Combined	.0129	**.0012**	.0032	.0026	.0042	.0023	.0122	.0036
L	**Aleatoric**	.0066	.0002	.0054	.0002	**.0001**	.0064	.0017	.0034
	Epistemic	.1700	.1586	.1970	.1860	.1633	.1766	**.1573**	.1719
	Combined	.0052	.0005	.0046	**.0004**	.0004	.0050	.0025	.0027
W	**Aleatoric**	.0108	.0011	.0051	**.0002**	.0009	.0111	.0055	.0009
	Epistemic	.1610	.1496	.1873	.1747	.1580	.1722	**.1365**	.1517
	Combined	.0084	.0004	.0043	.0004	**.0003**	.0088	.0070	.0008
2d box xmin	**Aleatoric**	**.0014**	.0049	.0323	.0747	.0202	.0036	.0280	.0073
	Epistemic	.0332	.0146	.0576	.0365	**.0004**	.0087	.0025	.0143
	Combined	**.0022**	.0145	.0352	.0790	.0383	.0113	.0440	.0149
2d box xmax	**Aleatoric**	**.0013**	.0062	.0241	.0639	.0110	.0017	.0306	.0084
	Epistemic	.0269	.0100	.0680	.0444	.0041	.0179	**.0001**	.0069
	Combined	**.0029**	.0176	.0269	.0683	.0263	.0061	.0493	.0180
2d box ymin	**Aleatoric**	.0147	.0022	**.0017**	.0127	.0210	.0038	.0276	.0060
	Epistemic	.0286	.0105	.0108	.0009	.0012	.0109	**.0003**	.0096
	Combined	**.0023**	.0024	.0074	.0300	.0373	.0107	.0456	.0143
2d box ymax	**Aleatoric**	.0045	**.0003**	.0103	.0385	.0121	.0019	.0372	.0105
	Epistemic	.0188	.0052	.0081	**.0002**	.0017	.0139	.0009	.0040
	Combined	**.0011**	.0097	.0192	.0554	.0292	.0072	.0578	.0217
Average MSE	**Aleatoric**	.0374	.0267	.0177	.0235	**.0104**	.0116	.0193	.0124
	Epistemic	.0831	.0662	.0897	.0743	**.0539**	.0686	.0616	.0746
	Combined	.0184	.0128	.0135	.0243	.0155	**.0096**	.0297	.0141

Table 4.4: All calibration Mean Squared Error results averaged over all 3D parameters for different models. For all evaluations the 2D scorer is used and the only filtering done is for Cars. This is evaluated for both combined uncertainties and for epistemic and aleatoric uncertainties separately. The model *UN F1 Gauss* evaluated as a Gaussian has the lowest average MSE for the aleatoric and epistemic uncertainties and the second lowest for the combined uncertainties.

Model / Uncertainty	UN F1 Lapl/Lapl	UN F1 Lapl/Gauss	UN Val Lapl/Lapl	UN Val Lapl/Gauss	UN F1 Gauss/Gauss	UN F1 Gauss/Lapl	UN Val Gauss/Gauss	UN Val Gauss/Lapl
Aleatoric	0.0556	0.0400	0.0181	0.0098	**0.0072**	0.0167	0.0128	0.0149
Epistemic	0.1152	0.0983	0.1202	0.1051	**0.0837**	0.1005	0.0962	0.1123
Combined	0.0277	0.0138	0.0086	**0.0050**	0.0057	0.0100	0.0168	0.0123

4.3.3 Uncertainty Error

A comparison between models, scoring functions, type of uncertainties and combinations of parameters is presented in Table 4.5 that contains minimum uncertainty errors (MUE) for these different combinations. What can be seen from the table is that the model with lowest (best) MUE does not differ between the different uncertainties (aleatoric, epistemic and combined) nor between the parameter sets. However, it appears to be the scoring function that determines what model will give the lowest uncertainty error. When using the 2D scoring method, *UN F1 Laplace* induces the lowest (best) MUE for all evaluated combinations of models and parameters. For the 3D scoring function *UN F1 Gauss* yields the lowest MUE. For the bird's-eye-view scoring function it is harder to determine what model yields the lowest minimum uncertainty error. However, *UN F1 Laplace* has the highest (worst) average MUE for the bird's-eye-view scoring function and the parameter configurations used in the table. Uncertainty Error plots for the parameters that are used to calculate the IoU for the bird's-eye-view scoring function (L, W, X, Z, θ) are visualized in Figure 4.29. Note that by only changing the scoring functions all of the detections will have the same uncertainty estimates and will therefore not change location in the plot, it is only the classification of the detections as true positives or false positives that might change when changing scoring function.

Table 4.5: Table containing minimum uncertainty error for different IoU scoring methods (2D, 3D and BEV) and uncertainties (Aleatoric, Epistemic and Combined). The model where a specific parameter set is best at separating true from false positives for a specific scorer and uncertainty, are highlighted in bold.

Minimum Uncertainty Error

Param. set	Scorer	Uncert.\Model	UN F1 Lapl.	UN Val Lapl.	UN F1 Gauss	UN Val Gauss
2D params: **2D BBOX**	**2D**	**Aleatoric**	**.380**	.404	.428	.465
		Epistemic	.382	**.360**	.453	.470
		Combined	**.385**	.409	.438	.471
	3D	**Aleatoric**	.351	.365	**.289**	.295
		Epistemic	.365	.340	**.306**	.311
		Combined	.358	.360	**.287**	.295
	BEV	**Aleatoric**	.411	.431	.393	**.327**
		Epistemic	.414	.419	.427	**.361**
		Combined	.409	.428	.401	**.329**
3D params: **Dimensions, Location, Rotation**	**2D**	**Aleatoric**	**.238**	.266	.252	.257
		Epistemic	**.262**	.273	.314	.334
		Combined	**.218**	.258	.256	.259
	3D	**Aleatoric**	.415	.433	**.393**	.425
		Epistemic	.449	.424	**.412**	.442
		Combined	.452	.427	**.390**	.431
	BEV	**Aleatoric**	.403	.359	**.353**	.367
		Epistemic	.404	**.367**	.392	.423
		Combined	.391	.360	**.355**	.372
BEV params: **Length, Width, X, Z, Rotation**	**2D**	**Aleatoric**	**.251**	.276	.262	.267
		Epistemic	**.251**	.268	.298	.326
		Combined	**.241**	.259	.266	.266
	3D	**Aleatoric**	.465	.423	**.397**	.422
		Epistemic	.443	.421	**.407**	.449
		Combined	.459	.415	**.395**	.437
	BEV	**Aleatoric**	.410	.371	.357	**.350**
		Epistemic	.419	**.365**	.386	.417
		Combined	.403	.364	**.359**	.371

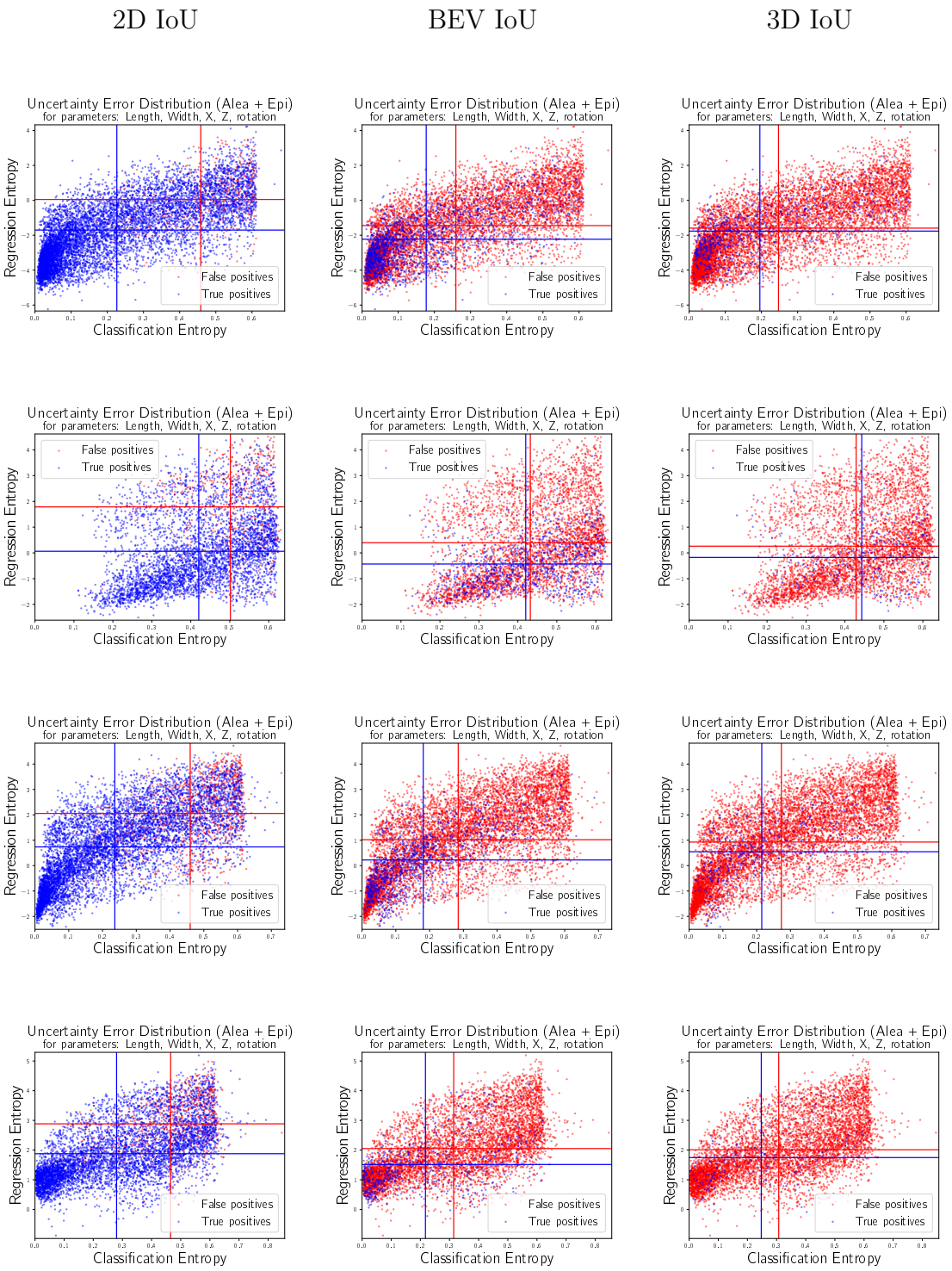


Figure 4.29: The Uncertainty Error plots for different models and scoring functions. The models are *UN F1 Laplace*, *UN Val Laplace*, *UN F1 Gauss* and *UN Val Gauss* from top to bottom and the scoring methods are 2D, Birds Eye View and 3D IoU from left to right. Note that by only changing scoring function (left to right) it is only the classification of the detections as true positives (blue) or false positives (red) that changes. All of the dots are located on exactly the same place.

4.4 Performance metrics

The performance metrics are used for evaluating the pure performance of the object detection task. This is evaluated for the different networks and the metrics used are *average precision* and *detection statistics*.

4.4.1 Average Precision

The metric used in the KITTI benchmark is Average Precision (AP), described in Section 2.5.5. AP is calculated for all networks, KITTI difficulty levels, different scoring functions and different corresponding scoring thresholds. AP is however, only presented for cars since the amount of cars in the data set is substantially larger. The results for the scoring functions 2D, 3D and bird's-eye-view (BEV) are presented in Tables 4.6, 4.7 and 4.8 respectively.

Table 4.6: Table containing 2D Average Precision values for cars for the different networks, scoring thresholds and KITTI splits.

Metric	**Average Precision**		
Scorer	**2D**		
Scoring threshold	**0.3**	**0.5**	**0.7**
Kitti Split	**E/M/H**	**E/M/H**	**E/M/H**
Baseline	.9081/**.9040**/.8122	.9054/**.8978**/**.8071**	.8965/**.7936**/**.7028**
Baseline + Homoscedastic	.9051/.8992/**.8767**	.9031/.8931/.8043	.8795/.7806/.6922
UN F1 Laplace	**.9090**/.8990/.8071	**.9088**/.8939/.7989	**.9001**/.7933/.6946
UN Val Laplace	.9043/.8380/.7604	.8941/.7711/.6890	.7930/.6075/.4877
UN F1 Gauss	**.9090**/.8851/.7978	.9082/.8605/.7730	.8820/.6795/.5837
UN Val Gauss	.9085/.8576/.7807	.9048/.7882/.7021	.8130/.5761/.4951

Table 4.7: Table containing 3D Average Precision values for cars for the different networks, scoring thresholds and KITTI splits.

Metric	**Average Precision**		
Scorer	**3D**		
Scoring threshold	**0.3**	**0.5**	**0.7**
Kitti Split	**E/M/H**	**E/M/H**	**E/M/H**
Baseline	.5035/.4493/.4009	.1927/.1597/.1318	.0051/.0071/.0080
Baseline + Homoscedastic	.4629/.4218/.3756	.1247/.1256/.1044	.0031/.0045/.0049
UN F1 Laplace	**.6979**/**.5259**/**.4586**	**.2595**/**.2108**/**.2034**	**.0111**/**.0084**/**.0093**
UN Val Laplace	.5066/.4013/.3472	.1390/.1129/.1137	.0030/.0044/.0052
UN F1 Gauss	.5730/.4295/.3660	.1600/.1275/.1243	.0035/.0046/.0052
UN Val Gauss	.3733/.3010/.2551	.1092/.0849/.0840	.0021/.0027/.0027

Table 4.8: Table containing Birds Eye View Average Precision values for cars for the different networks, scoring thresholds and KITTI splits.

Metric	**Average Precision**		
Scorer	**BEV**		
Scoring threshold	**0.3**	**0.5**	**0.7**
Kitti Split	**E/M/H**	**E/M/H**	**E/M/H**
Baseline	.5893/.4786/.4241	.2941/.2432/.2105	.0357/.0321/.0338
Baseline + Homoscedastic	.5504/.4643/.4100	.2612/.2298/.1936	.0266/.0255/.0250
UN F1 Laplace	**.7197/.5478/.4748**	**.4373/.3373/.2809**	**.0766/.0783/.0792**
UN Val Laplace	.5377/.4248/.3656	.2977/.2383/.2260	.0224/.0314/.0245
UN F1 Gauss	.6212/.4659/.3881	.2955/.2255/.1790	.0274/.0318/.0190
UN Val Gauss	.4504/.3345/.3116	.1715/.1412/.1329	.0205/.0194/.0202

4.4.2 Detection Statistics

The detection statics metric is a common way of representing the quantities of correctly and incorrectly detected objects as well as missed objects, i.e. true positives, false positives and false negatives (see Section 2.5.3). It is evaluated for all networks, KITTI difficulty levels and for the different scoring functions. The results can be seen in Tables 4.9, 4.10 and 4.11 corresponding to scoring function 2D, 3D and BEV.

Table 4.9: Table containing all 2D Detection Statistics for Cars for all different network setups for scoring threshold 0.7.

Metric	**Detection Statistics**		
Scorer	**2D**		
Kitti Split	**Easy**	**Moderate**	**Hard**
Number of	**TP/FP/FN**	**TP/FP/FN**	**TP/FP/FN**
Baseline	3104/1073/45	7615/2531/1106	9106/2531/3194
BL + Homosc.	**3110**/1628/**39**	**7673**/3454/**1048**	**9193**/3454/**3107**
UN F1 Laplace	3017/**310**/132	6406/503/2315	7369/503/4931
UN Val Laplace	2082/421/1067	3409/**476**/5312	3883/**476**/8417
UN F1 Gauss	2961/750/188	5693/1201/3028	6524/1201/5776
UN Val Gauss	2676/715/473	4439/799/4282	5065/799/7235

Table 4.10: Table containing all 3D Detection Statistics for Cars for all different network setups for scoring threshold 0.7.

Metric	**Detection Statistics**		
Scorer	**3D**		
Kitti Split	**Easy**	**Moderate**	**Hard**
Number of	**TP/FP/FN**	**TP/FP/FN**	**TP/FP/FN**
Baseline	297/8228/2852	573/12414/8148	669/12414/11631
BL + Homosc.	248/9335/2901	513/14006/8208	565/14006/11735
UN F1 Laplace	**335**/6206/**2814**	**610**/7991/**8111**	**700**/7991/**11600**
UN Val Laplace	149/**4381**/3000	229/**4500**/8492	269/**4500**/12031
UN F1 Gauss	192/6979/2957	327/8176/8394	377/8176/11923
UN Val Gauss	105/6230/3044	150/6500/8571	163/6500/12137

Table 4.11: Table containing all bird's-eye-view Detection Statistics for Cars for all different network setups for scoring threshold 0.7.

Metric	**Detection Statistics**		
Scorer	**BEV**		
Kitti Split	**Easy**	**Moderate**	**Hard**
Number of	**TP/FP/FN**	**TP/FP/FN**	**TP/FP/FN**
Baseline	682/7489/2467	1276/11536/7445	1477/11536/10823
BL + Homosc.	647/8632/2502	1222/13167/7499	1344/13167/10956
UN F1 Laplace	**819**/5276/**2330**	**1474**/6943/**7247**	**1663**/6943/**10637**
UN Val Laplace	477/**3755**/2672	751/**3873**/7970	841/**3873**/11459
UN F1 Gauss	626/6228/2523	983/7394/7738	1087/7394/11213
UN Val Gauss	393/5767/2756	578/6033/8143	619/6033/11681

5
Discussion

5.1 Results

In this section we will discuss the results presented in Chapter 4. Firstly discussions regarding the visualizations are presented. Secondly, the results for the uncertainties and the different metrics describing them are thoroughly discussed followed by finishing comments regarding the performance results.

In the KITTI benchmark, described in Section 2.5.1, today's standard way of classifying detections as either true positives (TPs) or as false positives (FPs) in object detection is used, i.e. that a detection should have (3D, BEV or 2D) IoU to a ground truth box over a certain threshold to be classified as a TP, otherwise it becomes a FP. This notation of a TP is based solely on the estimated mean boxes and is thus strongly tied to the pure performance of the network. In other words, our uncertainty estimates can, with today's way of distinguishing TPs from FPs *not* be used to increase performance, except for that it might help during training to attenuate for outliers which, by extension, might lead to that the network learns better. For a 3DOD machine learning algorithm to be used optimally in a safety critical application some notion of uncertainty is needed about the estimates. This motivates the need to find some other way of classifying detections into TPs and FPs to also include the uncertainties that the network indicates. By reformulating this, the uncertainty estimates could naturally be used to boost the performance of the network, thus yielding yet another way of analyzing the quality of the uncertainty estimates, in addition to the uncertainty metrics. This reformulation is further discussed in Section 5.3.1.

5.1.1 Visualizations

The top view figures with the joint uncertainty visualized (Figures 4.6, 4.8 and 4.10) in Section 4.1 clearly shows that some of the boxes, where the BEV IoU is not 0.7, as required for a TP in the KITTI benchmark, to a ground truth box is indeed covered within one standard deviation. In the sample images that we have looked at it is most often the Z estimate that is off and which causes the mean box to not overlap with the ground truth box enough to be classified as a TP, however, the network typically also indicates that it is uncertain by increasing the uncertainty estimate for these objects.

5.1.2 Calibration

In this section the calibration results for the UncertaintyNet stopped on highest F1 score and trained under the assumption of Laplacian observation noise corrupting the outputs (*UN F1 Laplace*) model will be discussed.

As briefly mentioned in Section 4.2.2 the model is consistently over-confident about its predictions for the 3D parameters. This can be seen in Figure 4.15 from that the calibration curves lies below the perfect calibration line $y = x$. From these figures it is quite clear that the output from the *UN F1 Laplace* model does *not* follow a Laplacian distribution characterized by the estimated mean and estimated variance. However, the outputs might still follow, for instance, a Laplacian distribution with some other variance or perhaps a Gaussian distribution. An initial step for analyzing if the outputs seem to follow some other distribution is done by treating the outputs from *UN F1 Laplace* as if they were to characterize a Gaussian distribution and evaluating the calibration under this assumption. This is discussed further in Section 5.1.3.

A final note here is that we do not expect the epistemic uncertainties to be calibrated as these uncertainties should, theoretically, vanish as we observe more data and become more certain about what model that generated our data. That we find a better approximation for the function that generated our data could, potentially, lead to that the distribution of our mean predictions becomes sharper and thus are closer to the ground truth targets at a higher rate. This could result in that our model becomes better calibrated as the aleatoric uncertainties are supposed to describe uncertainty in the data and we should not be able to reduce this uncertainty by observing more data.

5.1.3 Calibration comparison

In this section the comparison between models done in Section 4.3.2 will be discussed. First off the calibration for the length parameter was compared in Figure 4.27. This was done for two models that both stopped on highest F1 score but where the observation noise was assumed to be either Laplacian or Gaussian. The models were evaluated as if their predictions characterized first a Laplacian distribution and then a Gaussian distribution. A Laplacian distribution and a Gaussian distribution are quite similar in shape but the Laplacian probability density function (PDF) has more of its area closer to its mean and longer tails compared to a Gaussian distribution. The effect that this has can clearly be seen in Figure 4.27 as the same model evaluated as both Laplacian and Gaussian has similar calibration curves for probabilities higher than 0.8 and before that the Laplace-evaluation curve is consistently lower than the Gaussian-evaluation curve. This is, again, as the interval for which e.g. 20% of the area under the PDF is covered is at $\approx \pm 0.25\sigma$ for the Gaussian distribution compared to $\approx \pm 0.15\sigma$ for the Laplacian distribution. Which means that when σ is fixed the interval, for which the ground truth should lie within in 20% of the cases, is wider for the Gaussian distribution compared to the Laplacian distribution. The length parameter for both models stopped on F1 loss seems to be described by a Gaussian distribution characterized by the estimated mean and the estimated variance quite accurately, as seen in Figure 4.27, however this claim

would need further analysis to be established. It could also be that the length parameter for *UN F1 Laplace* can be accurately modelled by the estimated mean and the estimated variance if the variance were to be scaled by some constant value. However, no such experiments were conducted.

A parameter such as the length parameter, where the variance in the ground truth targets is small, the network will be able to learn this quite quick and thus quickly make the error to ground truth small. However, for a parameter where the variance in the parameter is large, as for the Z parameter, the network will not learn this as quick and thus the error to the ground truth target will remain larger for longer. When a model is trained under the assumption of Gaussian observation noise corrupting the outputs the loss function which the model is trained contains an L_2 norm as opposed to an L_1 norm for the Laplacian assumption, see Equation (2.10), (2.11) for details. When the error between the estimate and the ground truth data can be large, as for the Z variable, it might be troublesome to use an L_2 loss as the squared error will push the learnable σ to become large to reduce the total loss. This *might* be the reason as to why the models trained under the assumption of Gaussian observation noise are under-confident in Figure 4.28. Moreover, the Gaussian model stopped on lowest validation loss is more under-confident compared to the Gaussian model stopped on F1 score, which points towards the same conclusion as that model has been trained for substantially fewer epochs and yet the frequency for which the ground truth lies within certain multiples of σ is higher for that model, indicating that the network outputs larger uncertainties.

As discussed in Section 4.2.2 the outputs from the *UN F1 Laplace* model can not, accurately, be described by a Laplacian distribution characterized by the estimated mean and the estimated variance. However, from Table 4.3 we see that, for all the 3D parameters, the *UN F1 Laplace* model evaluated as Gaussian is better calibrated than the *UN F1 Laplace* model evaluated as Laplace. As these variables were over-confident as Laplacian distributions it is quite obvious, with the previous discussion about confidence intervals for the two distributions, that this would be the case. From these results further analysis is needed to conclude what distribution that most accurately describes the variable. Again, the model might become less over-confident by simply observing more data to make the predictions sharper whilst the aleatoric uncertainties should stay rather constant, or it could be that a simple scaling of the estimated variance could turn out to make the model well calibrated. We leave this analysis for future research.

5.1.4 Area Under Sparcification Error

In this section the Area Under Sparcification Error results for the UncertaintyNet stopped on highest F1 score and trained under the assumption of, in short the *UN F1 Laplace*, model will be discussed.

What can be observed in Section 4.2.1 is that length, width and Y have the highest and Z, depth, 2DBBOX xmin and 2DBBOX xmax the lowest values for Area Under Sparcification Error. The sparcification curves for the length parameters (see Figure 4.12) are flat and sometimes even have a positive derivative. This indicates that network has no ability to sort its uncertainties according to actual error for the

length parameter and could be compared to a homoscedastic uncertainty being a constant uncertainty for a parameter instead of varying with the input. The reason for this *might* be due to that uncertainties about these parameters are in general of small magnitude and the variance of the uncertainty estimates is also quite small (See uncertainties for length and width in the top view visualization in Figure 4.5). However, even though the uncertainty estimates for these parameters seem to make no sense when evaluating AUSE the uncertainty estimates may still be valuable (e.g. in distinguishing TPs from FPs) and well calibrated even though they are generally unsorted.

In Figure 4.11 the sparcification curves for the rotation parameter is visualized. The curve for the aleatoric uncertainty initiates with a negative derivative in the start, but turns exponential towards the end where a big fraction of the detections have been removed. This appear to happen for aleatoric but not to epistemic uncertainty to the same extent. One idea behind this phenomena is that the network might be estimating very small aleatoric uncertainties for objects turned 180 degrees, resulting in a large error to ground truth but yet small uncertainties. This would then cause a big increase in average error when the flipped estimates remains with only a few other detections towards the end of the curve. This phenomena has not been extensively analyzed and further analysis is required to establish the reason for it.

To draw some general conclusion regarding the Area Under Sparcification Error and different parameters one might argue that parameters with larger values that varies substantially, such as Z, are better at estimating well sorted uncertainties, i.e. resulting in a lower AUSE value, than parameters with opposite qualities. The uncertainties for variables like the dimension variables, where the uncertainties are typically small, results in poorly sorted uncertainty estimates and thus a high AUSE. However, this is probably due to noise in the uncertainties rather than that the uncertainty predictions are bad, as the dimension parameters are generally more calibrated than some other parameters which have a lower AUSE. This highlights one of the aspects where the evaluation metrics are inadequate on their own but meaningful in combination.

5.1.5 Area Under Sparcification Error comparison

From the comparison between the different models with respect to Area Under Sparcification Error, what may concluded is that there is no substantial general difference between the models. The parameters that the different models are good at sorting in the order of actual error varies quite much. *UN F1 Laplace* is good for Z and 2D bounding box and *UN Val Laplace* is exceptionally good at rotation. In Table 4.2 we see that overall the Laplacian models have a lower AUSE compared to the Gaussian models.

As discussed in Section 5.1.1, no comprehensive evaluation of what parameter that affects performance metrics the most has been done, but based on empirical observations the Z parameters appears to do so. One could, therefore, argue that *UN F1 Laplace* would be the preferred model with respect to AUSE performance as that model has quite well-sorted uncertainties for the Z parameter.

5.1.6 Uncertainty Error

In this section the uncertainty error results for the UncertaintyNet stopped on highest F1 score and trained under the assumption of Laplacian observation noise corrupting the outputs (in short, the *UN F1 Laplace*) model will be discussed.

What can be concluded from the results in Section 4.2.3 is that, since the Minimum Uncertainty Error never reaches zero, the model is not able to distinguish TPs from FPs by the uncertainty estimates. If a threshold, for which the uncertainty estimates can differentiate TPs from FPs, would be found then an extra step in filtering detections could be added. This filtering step would require all of the detections that pass the classification threshold to also have an uncertainty entropy below this threshold to be accepted as an output. If this further filtering of outputs would be flawless it would result in zero false positives, which is extremely valuable in a safety critical system where actions based on false detections could be unpleasant and in some situations dangerous.

From Table 4.5 in Section 4.3.3 we see that the lowest Minimum Uncertainty Error is approximately 0.22. It is hard to tell whether this is a good value or not, with zero being the lowest achievable value and 0.5 being the worst possible MUE, 0.22 is slightly below the middle. The following toy example might give some intuition as to what a MUE value of 0.22 *might* mean. Pose that the MUE value is reached for an entropy threshold for which all false positives are above, then 44 % of all true positives would also be above this threshold. This toy example can be calculated as

$$0.22 = \text{MUE}(\delta) = \frac{1}{2}\left(\frac{\#\text{TP} > \delta}{\#\text{TP}} + \frac{\#\text{FP} < \delta}{\#\text{FP}}\right) = \left\{\frac{\#\text{FP} < \delta}{\#\text{FP}} = 0\right\} = \frac{1}{2}\frac{\#\text{TP} > \delta}{\#\text{TP}} \Rightarrow$$
$$\Rightarrow 0.44 = \frac{\#\text{TP} > \delta}{\#\text{TP}} \;\Rightarrow\; 44\% \text{ of all TPs have entropy above } \delta$$

where δ is the entropy threshold.

For the different scoring functions used one might think that the parameters that are used to calculate the IoU for the specific scorer would be the best parameters to discriminate TPs from FPs. However, this is not the case for the MUEs presented in Table 4.1. For instance the uncertainties for the Y world coordinate is the parameter that best discriminates TPs from FPs for the 2D scoring function. This *might* be since the uncertainty for Y is derived from uncertainties in the 2d bounding box's upper and lower edge.

An example is used to reason as to why the uncertainties in Y tell us more about false positives for the 2D scoring function compared to what the uncertainties for the 2D bounding box estimates tell us. Pose that the network detects an object close to the camera. The network might not be able to distinguish at exactly what pixel the object's top and bottom edge, respectively, is located, and for objects close to the camera the edge itself might cover multiple pixels. However, it is highly unlikely that the network would not be able to predict a 2D bounding box that have a certain 2D IoU overlap with the ground truth box, even if its uncertainty about the edges might be quite big (many pixels). This uncertainty is then unprojected into the world but as the object is close this big uncertainty about the bounding box edges will be translated to small uncertainties about the Y coordinate as each

pixel, for objects this close, might correspond to centimeters in world coordinates. If the network were to predict the 2D bounding box for an object that is far away it would probably be easier to determine at what pixel the object's edges are located, and the estimated uncertainties might only be a few pixels. However, when these pixels are unprojected into the world the small difference in pixels could correspond to much larger distances in meters. Moreover, as objects far away look small in the image the area to calculate the 2D IoU with is drastically smaller. Thus a few pixels off might be enough for the IoU to become too small to be accepted as a TP. All of this reasoning boils down to that the uncertainties for the Y coordinate actually contains some information about how far away the object is located. This is as it is propagated from the 2D bounding box uncertainties, through the camera model and the resulting rays are cut at the Z distance to the object.

For the UN F1 Laplace model and the KITTI easy split a change of scoring function from 2D to 3D or BEV results in that the number of TPs drops from 3110 to 335 and 819, respectively, as can be seen in Tables 4.9, 4.10 and 4.11. We see from Table 4.1 that the X parameter is best at distinguishing TPs from FPs for the BEV scoring function and that the x and y 2D bounding box edges best distinguishes TPs from FPs for the 3D scoring function. Again, the uncertainty in X is propagated from uncertainty in the 2D bounding box edges. Given the drastic drop in TPs for the 3D and BEV scoring functions a lot of relatively close objects (most close to semi-close objects are TPs for 2D scorer, reasoning above) have now become FPs. As the 3D center point is derived from the 2D bounding box and the depth estimate a slight offset of the 2D bounding box will result in that the center point is not in the center of the object. This offset does not seem to matter when calculating IoU in the image plane but in 3D this offset has a much bigger impact. Furthermore, if the depth estimate is slightly off this 2D box offset might easily result in that the 3D or BEV IoU drops below 0.7, which is the required IoU for a detection to become a TP. Therefore, it *might* be more important for the 3D and BEV scoring functions that we are certain about the location of the 2D bounding box compared to when using the 2D scoring function.

The above reasoning can be summarized in a conclusion that the uncertainty estimates induce valuable information that can be used to distinguish true from false positives to some extent. In other words, the results from the UE indicate that there is a correlation between the uncertainty estimates and if the object is a TP or a FP. This information could, for example, be used in a post processing step to further remove detections based on their regression parameter uncertainties rather than only accepting or dismissing detections based on the classification score.

5.1.7 Uncertainty Error comparison

As described in Section 4.3.3 it appears to be the scoring function alone that determines what model will give the lowest uncertainty error. When using the 2D scoring method, *UN F1 Laplace* induces the lowest (best) MUE for all evaluated combinations of models and parameters. For the 3D scoring function *UN F1 Gauss* yields the lowest MUE. For the bird's-eye-view scoring function it is not as clear as to what model that performs the best MUE-wise. However, *UN F1 Laplace* has

the highest (worst) average MUE for the bird's-eye-view scoring function and the parameter configurations used in Table 4.5.

What might be concluded from this is that the *UN F1 Laplace* model is a good choice of model for the task of 3D Object Detection with uncertainty estimates. The reason for that is that, as was discussed in Section 5.1.1, with the accompanied uncertainties our detections might be able to cover objects that the pure mean would not be able to. This applies especially for the bird's-eye-view and 3D scenarios where the number of true positives drop substantially in comparison to 2D, which may be seen in the Detection Statistics Section 4.4.2. Therefore it is of big interest that these true positives remain true even if the estimated 3D mean might be a bit off also since the rate of false positives are low for *UN F1 Laplace* in 2D.

5.1.8 Uncertainty summary

In this section the discussions regarding the specific metrics will be merged into one brief discussion. For the *UN F1 Laplace* model neither of the outputs were particularly calibrated compared to other models. However, when considering all the parameters this model has the lowest (best) average AUSE, meaning that, on average, this model is best at sorting the uncertainties according to their error to ground truth. Furthermore, the uncertainty estimates can be used to, to some extent, discriminate TPs from FPs according to the UE metric.

The *UN Val Laplace* model appears to both be good in AUSE and calibration for the rotation parameter, thus indicating that this parameter is actually well learned. The arguably more valuable Z parameter is, however, better AUSE-wise for *UN F1 Laplace*, even though it is not especially calibrated. There is the possibility to calibrate a model in a post processing step, but how one would go about to lower the Area Under Sparcification Error in a post processing step is not obvious.

5.1.9 Performance

In this section the results from the performance metrics for the different models will be discussed. The metrics discussed are Average Precision (AP) and Detection Statistics.

The results presented in the performance Section 4.4 indicate that the uncertainty estimates does not affect the performance negatively, but rather boosts the performance. For the task of 2D object detection the AP for the baseline model and the models with uncertainties are comparable. However, for the harder task of 3D object detection the uncertainties seem to boost the pure performance of the model, possibly due to the ability for the network to attenuate for outliers during training and thus find a better local minimum.

As can be seen in 4.7 the AP for our model increases from about 1% to about 70% for the KITTI easy split when the 3D IoU threshold for a TP is lowered from 0.7 to 0.3. This further demonstrates that our mean predictions are fairly close to the ground truth boxes and that a quite small translation or rotation of the boxes would cause them to overlap with the GT boxes. Arguably, decreasing the IoU threshold can be compared to what would happen if we account for the uncertainties in the

predictions. Thus it is very likely that we would cover substantially more objects than what we do without the uncertainties.

A decrease in IoU threshold results in that a number of objects that were previously FPs become TPs. One may argue that accounting for the uncertainties would result in that at least the boxes that went from FPs to TPs would be covered.

The volume for which the estimated 3D box must lie within when decreasing the IoU threshold is an expanded version of the original threshold volume in all directions. That fact that our uncertainty estimates are, reasonably, calibrated means that the network can estimate within what interval the GT lies. Thus, instead of lowering the IoU threshold, and therefore expanding the volume that the estimated box must lie within in all directions, the original threshold volume can be preserved and the estimated box can be moved in the directions where we are uncertain. This way the estimated box should lie within the original threshold volume, and thus be classified as a TP, at some point. Arguably, this should be true in at least the cases where an IoU threshold decrease causes the estimated box to become a TP.

5.2 Approximations

In this section some of the approximations made in the project will be discussed. The approximation that will be discussed are propagation of uncertainties, potential correlation between estimates, world coordinate parameter estimation and pretrained gaussian models.

5.2.1 Propagation of uncertainties

The final 3D parameters were not estimated right way from the network but rather derived from estimates that lives closer to the raw data. This means that the estimated uncertainties have to also be propagated from the network output to the final parameters. The analytical expressions for how the uncertainties propagate from some variables to others could have been derived. However, this would require extensive derivations and ultimately computing power. Approximations were used instead to allow for more time to be spent on analysis of the outputs.

The uncertainty for the X and Y parameter is derived through the pseudo-inverse of the camera matrix to obtain two rays in the world. These rays were cut at the Z estimate and the variance in the X and Y direction was calculated, respectively. As the rays forms a cone in the world the variance in X and Y depends extensively on the depth estimate, which in itself is uncertain. Another way of propagating the uncertainties would be to cut the rays not only at the Z estimate but also perhaps at one standard deviation away from the Z estimate, $Z^{\pm} = Z \pm \sigma_z$, and then calculate some mean variance in X and Y respectively. This way uncertainty about the depth Z would be taken into account.

5.2.2 Potential correlation between estimates

The variables are assumed to be independent of each other, However, some of them are clearly not independent. One example of this is the actual width of a car (in

meters) and the width of the 2D bounding box in the image plane. There should, in fact, be a strong correlation between these estimates depending on how the car is oriented. An alternate way of choosing network outputs to not have this correlation would be to estimate the center point of the object in the image rather than the 2D bounding box. However, we decided to still estimate the 2D bounding box to be able to use the 2D scoring function as a scoring function in our experiments.

Also the X and Y world coordinates is correlated to the depth estimate by the way we propagate the information from the image plane to world coordinates. If the world coordinates are not to be estimated directly from the network this is inescapable due to the scale ambiguity that monocular vision algorithms suffers from.

5.2.3 World coordinate parameter estimation

With the network structure used in this project a mix of parameters belonging to the image plane and the world coordinates are estimated. Examples of parameters in the image plane are the 2D bounding box parameters and examples of parameters in the world coordinates are Z, dimensions and orientation. There is no answer to whether it is better to use a mix of parameters or stick to either parameters from the image plane or in world coordinates. For the task of 3D object detection with accompanying uncertainty estimates the need for conversion from image plane parameters to world coordinate parameters could be discarded. Since the conversion, at least for the uncertainties, induce approximations it would be interesting to see the results from estimating world coordinates straight away. One potential drawback, regarding immediate world coordinate parameter estimation, is that the world coordinate estimates are less connected to the raw data, i.e. an image, resulting in that the network might have to learn a more complex function.

5.2.4 Pretrained Gaussian models

One problem that we experienced with the Gaussian models was that the L_2 loss caused problems quite rapidly as some errors between estimate and GT can be quite big (2D bounding box, depth, X, Y) and by applying an L_2 loss functions they become even bigger. The network simply increased the aleatoric uncertainties drastically to attenuate for the big error, thus preventing, to some extent, the network from learning what it was supposed to. This could potentially be fixed by training the models without the uncertainty estimates for a number of epochs to allow the network to reduce the errors to make the L_2 loss reasonable small before adding the uncertainty estimates. Even if this would make the model perform better it is not clear what it would imply theoretically as the original loss function is derived from the negative log likelihood of a Gaussian distribution, it could potentially be reasoned to be yet another approximation.

5.3 Future work

In this section topics for future research is suggested. The topics handle incorporation of uncertainties in performance benchmarks, calibration post processing step, replacement of non-maximum-suppression, correlation between uncertainties and training data, active learning, performance boost using entropy and different methods for uncertainty extraction.

5.3.1 Framework to incorporate uncertainties in performance benchmarks

In this project uncertainty estimates have been added to the predicted mean value in object detection. The common approach is to work towards a network outputting flawless predictions. However, reaching perfection is tedious to both do and prove. Uncertainties are a clever way of working with an opposite approach. Instead of improving performance to perfection uncertainties give information about what the network does not yet know. An imperfect 3DOD network with perfect uncertainty estimations would, in theory, be able to provide sufficient information for being applied in a safety critical system, such as an autonomous vehicle. Perfect uncertainty estimates are also tedious to both do and prove, but how well does an imperfect network and its accompanied imperfect uncertainty estimates perform? Currently there is no way of evaluating this, but we strongly believe that it is the next step forward.

Firstly a method for determining the joint uncertainty volume for an object in which it with 68% certainty is inside needs to be produced. This joint volume would be so much more complex than a simple 3D box in its most accurate layout. An initial idea for creating it is to simply scale, rotate and translate the mean 3D box into a joint volume, similarly to what is done in this project for visualizing joint uncertainties.

Currently 3D IoU is used to distinguish true from false positives. But to simply compare the 3D IoU of the joint volume to the ground truth 3D box with this volume is not appropriate as the union between the two volumes would, probably, be too big to achieve an IoU above the TP threshold. To compare only the intersection of the volume with the ground truth box is also troublesome as by simply increasing the uncertainties to be infinite then all of the ground truth boxes could easily be covered, resulting in perfect recall. Therefore, a new scoring function needs to be created in which the above joint uncertainty volume gets top score for including the object volume and being as small as possible simultaneously.

Once this is done Average Precision may be evaluated for this new determination of true versus false positives. The ability to calculate detection statics and find out precision and recall would have been available. The entire process would result in a good framework for performance benchmarking of models with incorporated uncertainties for 3DOD.

5.3.2 Calibration post processing step

As discussed in Section 5.1.2 no further analysis of the distribution affiliation for the uncertainties can be made, since they are not calibrated to be of the correct magnitude. If all parameters would have been calibrated into having the minimum mean squared error to the perfect calibration line ($y = x$) then it would be possible to compare which one of the Laplacian or Gaussian distribution that the specific uncertainty follows best. By implementing a calibration post processing step this could be achieved. Calibration might be reached by simply multiplying each uncertainty with a task specific constant set to reach calibration.

5.3.3 Replace non-maximum-suppression with clustering algorithm

A lot of information is thrown away in the used non-maximum-suppression (NMS) algorithm presented in 2.4.4. It would be interesting to see if this information could be used to instead update mean and variance of estimates, to obtain some measure of epistemic uncertainty in a single forward pass as suggested in the *BayesOD* paper by Harakeh et. al [10]. The estimates in each pixel is instead seen as a measurement of the same variables. The variance derived from these pixel estimates does not necessarily account for all of the epistemic uncertainty, since the estimates would be from one forward pass with one set of network weights. If NMS is to be used, additional information could be used to distinguish, more accurately, what detections that are likely to correspond to the same object compared to when using only the 2D bounding box. For example, by also including the depth estimate in the NMS algorithm it is likely that the correct detection that was suppressed in the example seen in Figures 2.4 and 2.4 would not be suppressed.

5.3.4 Correlation between uncertainties and rate of which objects occurs in training data

As the uncertainties are obtained from a neural network that has learned from training data there is a possibility that there is a strong correlation between the number of training samples with a specific attribute and the estimated uncertainty. For example, the uncertainty for the depth estimates might have a strong correlation with the number of training examples at different depths. This could be examined by e.g. creating a number of bins (0-10m, 10-20m, etc) and comparing the average uncertainty for objects within a bin with the number of objects in each bin. These type of experiments could give further insights into what the network has learned and about how to improve the training data set.

5.3.5 Active Learning

For the subject of active learning, the induced uncertainty estimations might be used for finding cases where the network indicates high epistemic uncertainty. Since epistemic uncertainty are uncertainties due to imperfections in the model, objects

with high epistemic uncertainty are likely to be from scenarios that the network is not well trained for. Extending the training data set with images from such scenarios might have a much larger impact on the learning than by adding random images to the training data. Since annotation of images is time consuming, and thus expensive, this could be a cost effective and sophisticated way of improving the network performance and data set coverage.

5.3.6 Average Precision for detection split based on entropy

As discussed in Section 5.1.6 the entropy threshold derived for finding the minimum uncertainty error constitutes the threshold where false positives are most efficiently separated from true positives. It would be very interesting to extract the specific threshold for different sets of parameters, dismiss all detections above this threshold and evaluate how this affects the performance, i.e. Average Precision or Detection statistics.

5.3.7 Evaluate different methods for estimating uncertainties in 3DOD

Aleatoric can be modelled in real time as it only requires the network to estimate twice as many regression parameters, however, the way we model epistemic uncertainty is probably too expensive in real time and thus some other way of modelling this has to be found. Ilg et. al. present a network producing multiple hypotheses in a single forward pass which would have been interesting to evaluate[9]. Furthermore, as discussed briefly in Section 5.3.3, epistemic uncertainty could perhaps be modelled by simply replacing the NMS algorithm with some clustering algorithm as the different estimates for the same object might differ, thus indicating that the network is uncertain about the detection.

6
Conclusion

In this thesis, we studied two methods for estimating uncertainties in machine learning and applied them to a machine learning algorithm for monocular 3D object detection to enable per prediction uncertainty estimates. Furthermore, a method for propagating uncertainties to the global coordinate system was presented.

We show that a machine learning algorithm for monocular 3D Object Detection can successfully estimate the uncertainties in its predictions. Furthermore, a framework for evaluating uncertainties in 3D Object detection machine learning algorithms was proposed constituting of multiple complementary evaluation metrics. The different evaluation metrics give valuable information regarding the quality of the uncertainties in different aspects and are thus meaningful in combination but inadequate on their own. We reasoned, and gave intuitions about what the different metrics tell us for different parameters and scoring functions, to be able to compare models against each other. Lastly, we motivated as to why a new way of determining true positives is needed to be able to further evaluate the performance gain and usefulness of the uncertainty estimates in a safety critical system.

Bibliography

[1] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[2] T. Nair, D. Precup, D. L. Arnold, and T. Arbel, "Exploring Uncertainty Measures in Deep Networks for Multiple Sclerosis Lesion Detection and Segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Cham: Springer International Publishing, 2018, pp. 655–663.

[3] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 6 2016, pp. 1050–1059.

[4] Y. Gal, J. Hron, and A. Kendall, "Concrete Dropout," in *Advances in Neural Information Processing Systems*, 2017, pp. 3581–3590.

[5] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1321–1330.

[6] Y. Gal and Z. Ghahramani, "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference," *arXiv preprint arXiv:1506.02158*, 6 2015.

[7] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

[8] J. Kybic and C. Nieuwenhuis, "Bootstrap optical flow confidence and uncertainty measure," *Computer Vision and Image Understanding*, vol. 115, pp. 1449–1462, 2011.

[9] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, "Uncertainty estimates and multi-hypotheses networks for optical flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 652–667.

[10] A. Harakeh, M. Smart, and S. L. Waslander, "BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors," *arXiv preprint arXiv:1903.03838*, 2019.

[11] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[12] A. D. Kiureghian and O. Ditlevsen, "Aleatory or epistemic? Does it matter?" *Structural Safety*, vol. 31, no. 2, pp. 105–112, 2009.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[14] A. Graves, "Practical Variational Inference for Neural Networks," in *Advances in neural information processing systems*. Curran Associates, Inc., 2011, pp. 2348–2356.

[15] C. Blundell, J. Cornebise, K. Kavukcuoglu, W. Com, and G. Deepmind, "Weight Uncertainty in Neural Networks Daan Wierstra," *arXiv preprint arXiv:1505.05424*, 2015.

[16] Y. Gal, "Uncertainty in Deep Learning," Ph.D. dissertation, University of Cambridge, 2016.

[17] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*. IEEE, 1994, pp. 55–60.

[18] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace Distribution and Generalizations*. Springer Science+Business Media, LCC, 2001.

[19] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams under Orthography: a Factorization Method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[20] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 11 2004.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[22] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2011. [Online]. Available: http://dx.doi.org/10.1007/978-1-84882-935-0

[23] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision, Second Edition*, 2nd ed. Cambridge University Press, 2003.

[24] B. Li, "3D Fully Convolutional Network for Vehicle Detection in Point Cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1513–1518.

[25] B. Li, T. Zhang, and T. Xia, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," *arXiv preprint arXiv:1608.07916*, 2016.

[26] B. Tekin, S. N. Sinha, and P. Fua, "Real-Time Seamless Single Shot 6D Object

Pose Prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[27] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proceedings of the IEEE International Conference on Computer Vision*, 11 2017, pp. 1521–1529.

[28] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D Bounding Box Estimation Using Deep Learning and Geometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12 2017, pp. 7074–7082.

[29] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in neural information processing systems*, 6 2015, pp. 91–99.

[31] A. Krishnan and J. Larsson, *Vehicle detection and road scene segmentation using deep learning, MSc thesis*. Institutionen för signaler och system, Chalmers tekniska högskola, no: EX029/2016, 2016.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[33] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[34] 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, 2010.

[35] A. Rosebrock, "A visual equation for Intersection over Union (Jaccard Index)," 2016. [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[36] Walber, "Precision and recall," 2014. [Online]. Available: https://commons.wikimedia.org/wiki/File:Precisionrecall.svg

[37] D. Miller, F. Dayoub, M. Milford, and N. Sünderhauf, "Evaluating Merging Strategies for Sampling-based Uncertainty Techniques in Object Detection," *arXiv preprint arXiv:1809.06006*, 2018.

[38] E. Jörgensen, C. Zach, and F. Kahl, "Monocular 3D Object Detection and Box Fitting Trained End-to-End Using Intersection-over-Union Loss," *arXiv preprint arXiv:1906.08070*, 2019.

[39] F. Yu, V. Koltun, and T. Funkhouser, "Dilated Residual Networks," in *Pro-*

ceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 472–480.

[40] J. S. Milton and J. C. Arnold, *Introduction to Probability and Statistics*. McGraw-Hill Professional, 2002.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

A

Appendix 1

This appendix contains visualizations of the 3D parameters for the uncertainty metrics, calibration, area under sparcification error and uncertainty error. A number of additional parameter combinations are presented for the uncertainty error metric. The plots further visualize the metrics for the four main models used in this project.

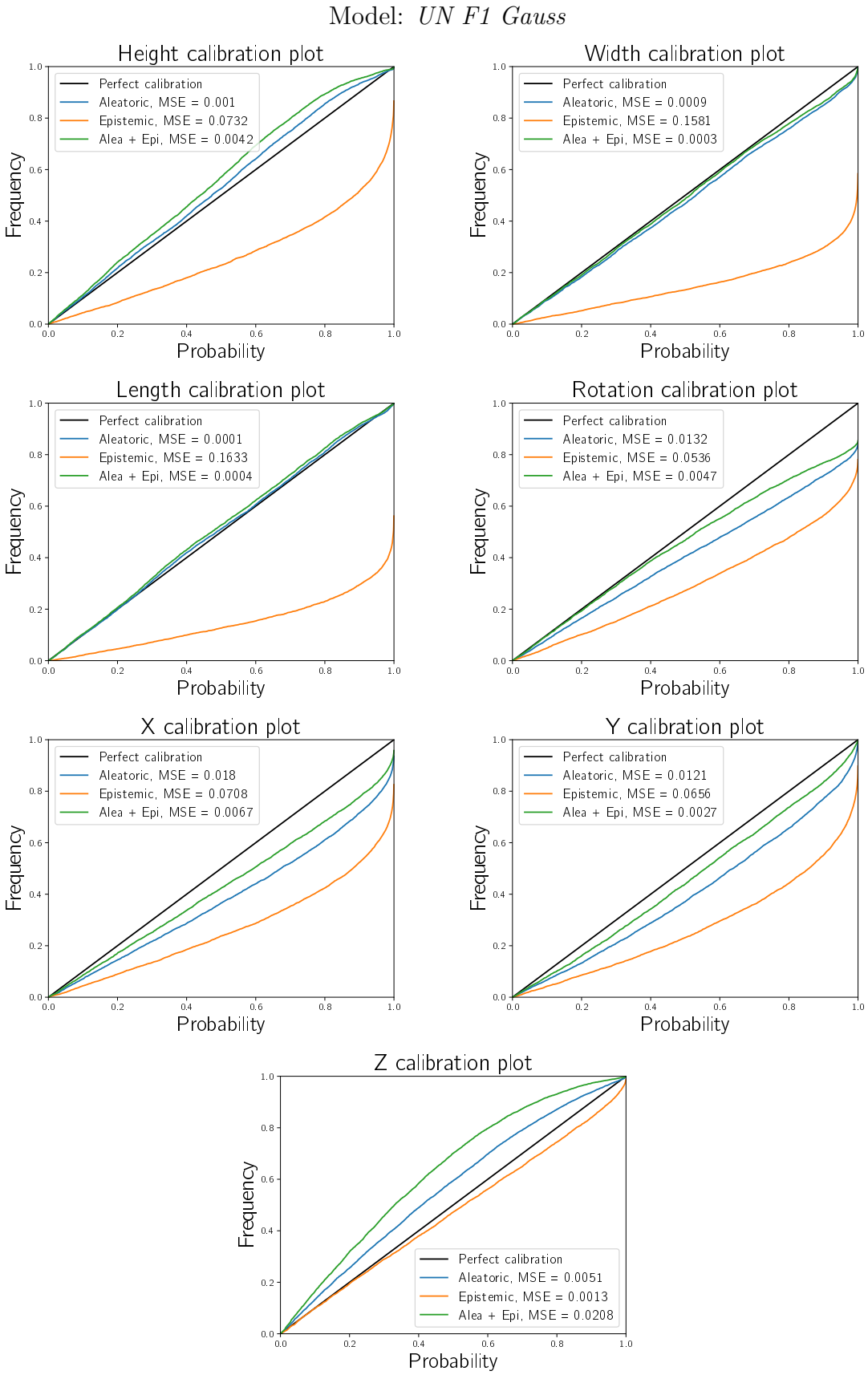


Figure A.1: Calibration plots of all the 3D variables for the model *UN F1 Gauss*.

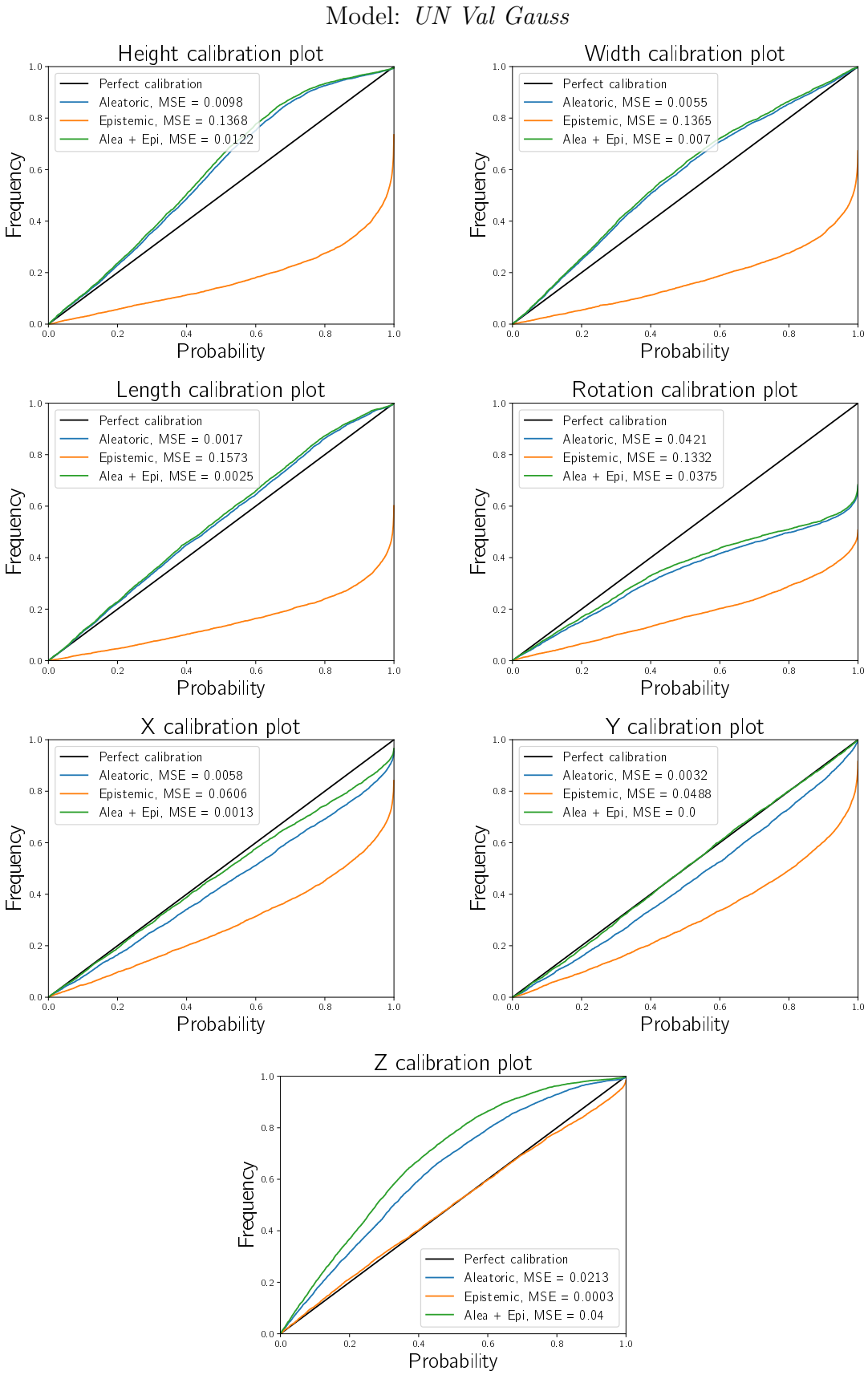


Figure A.2: Calibration plots of all the 3D variables for the model *UN Val Gauss*.

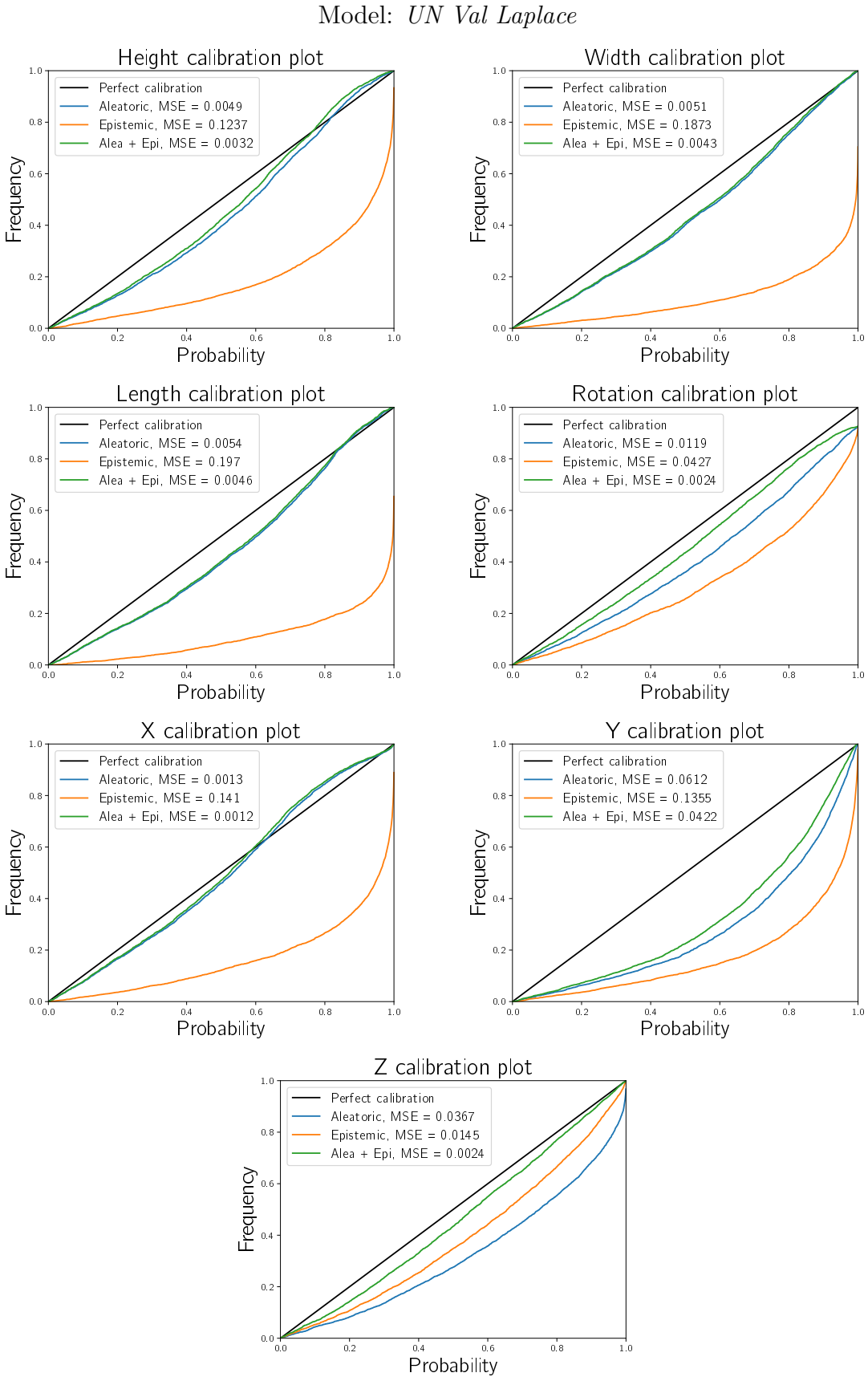


Figure A.3: Calibration plots of all the 3D variables for the model *UN Val Laplace*.

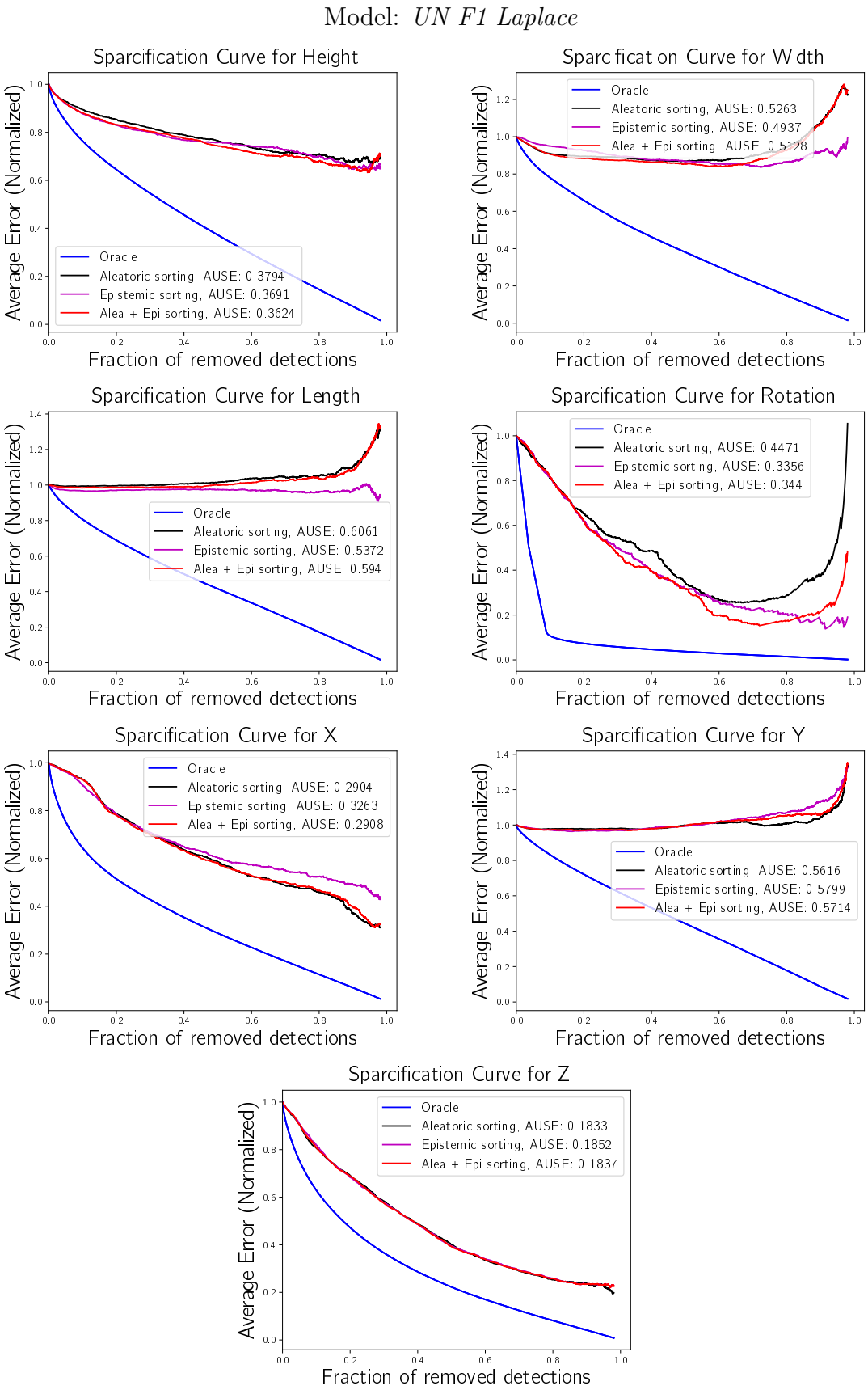


Figure A.4: Sparcification Curve plots of all the 3D variables for the model *UN F1 Laplace*.

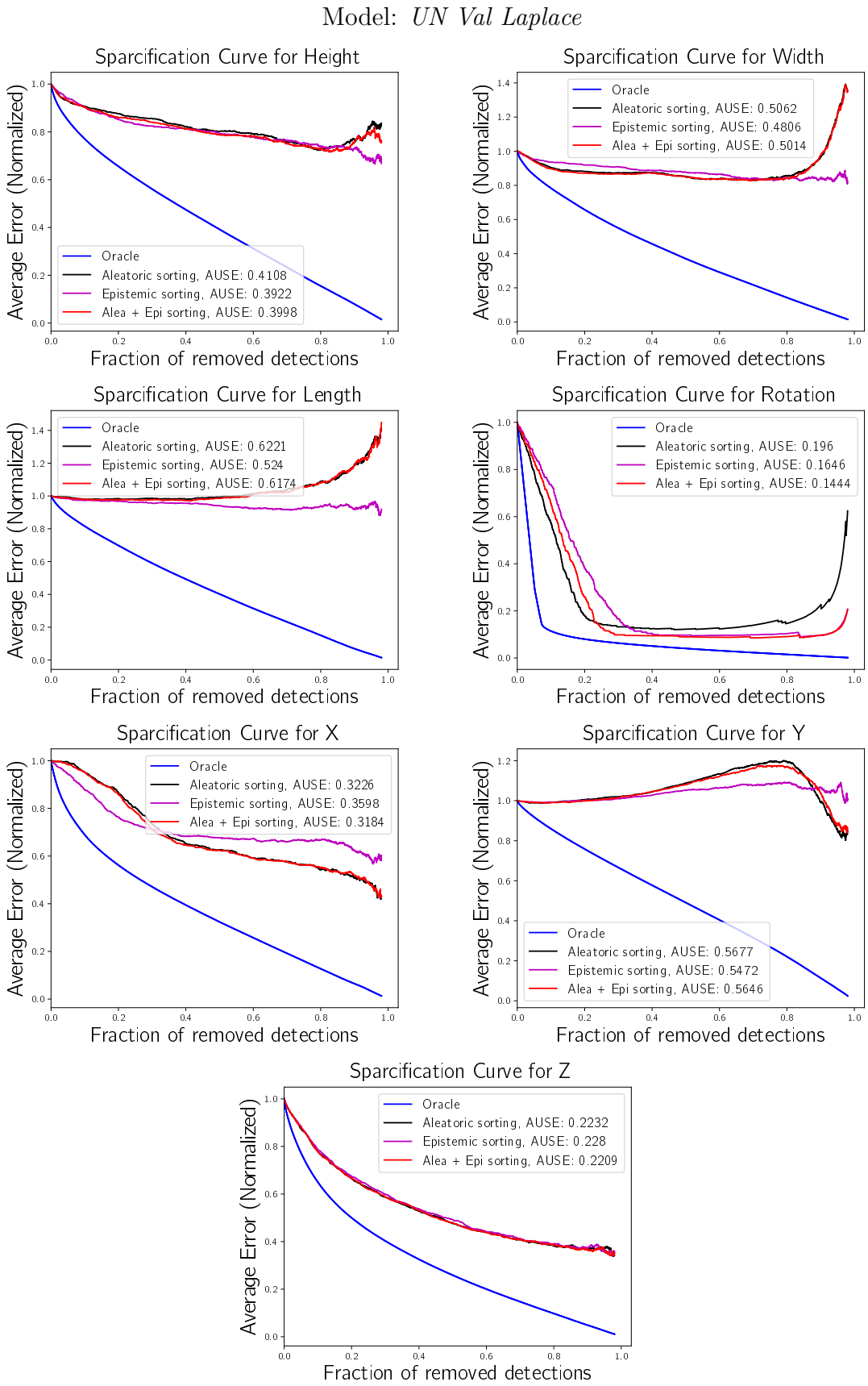


Figure A.5: Sparcification Curve plots of all the 3D variables for the model *UN Val Laplace*.

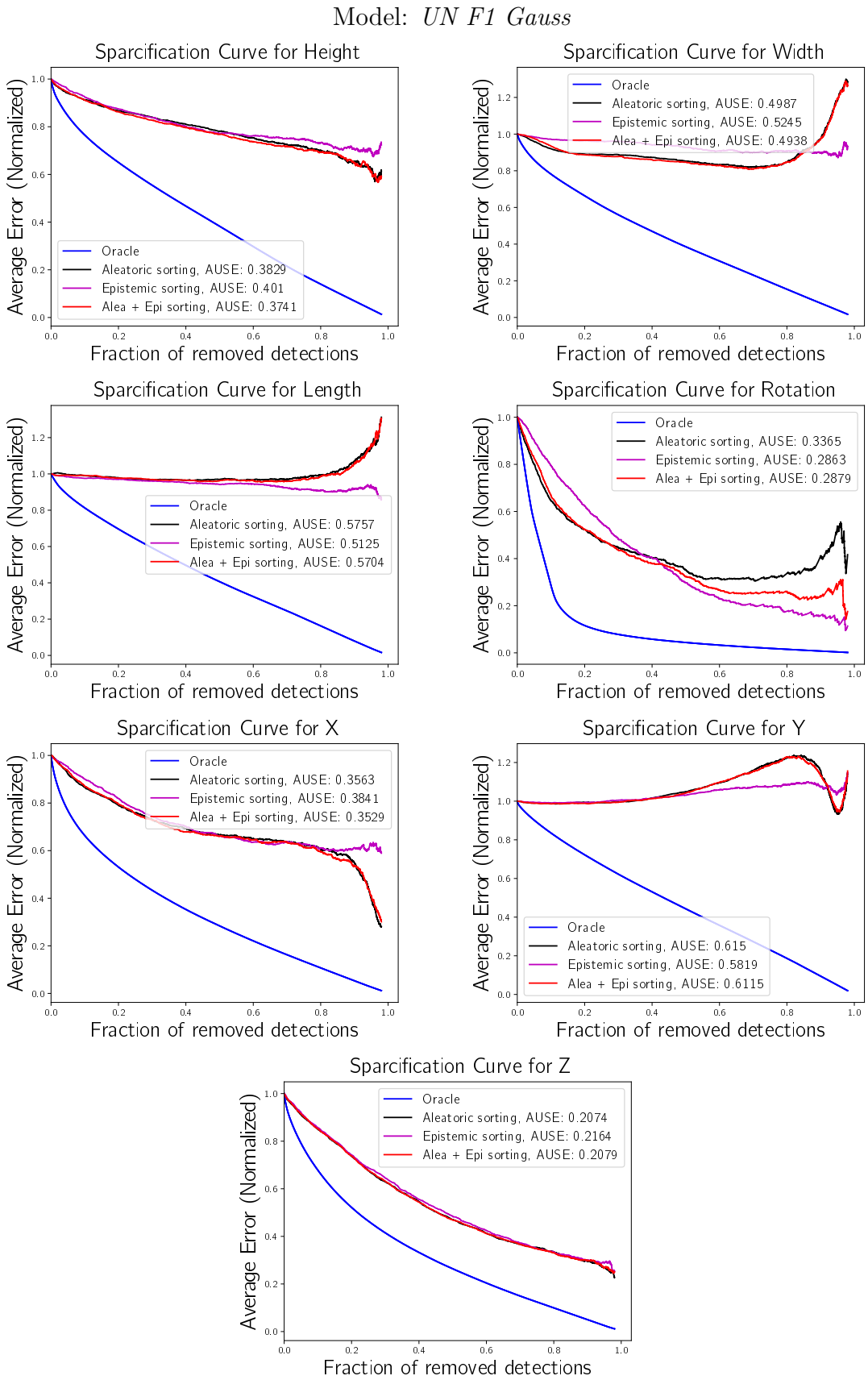


Figure A.6: Sparcification Curve plots of all the 3D variables for the model *UN F1 Gauss*.

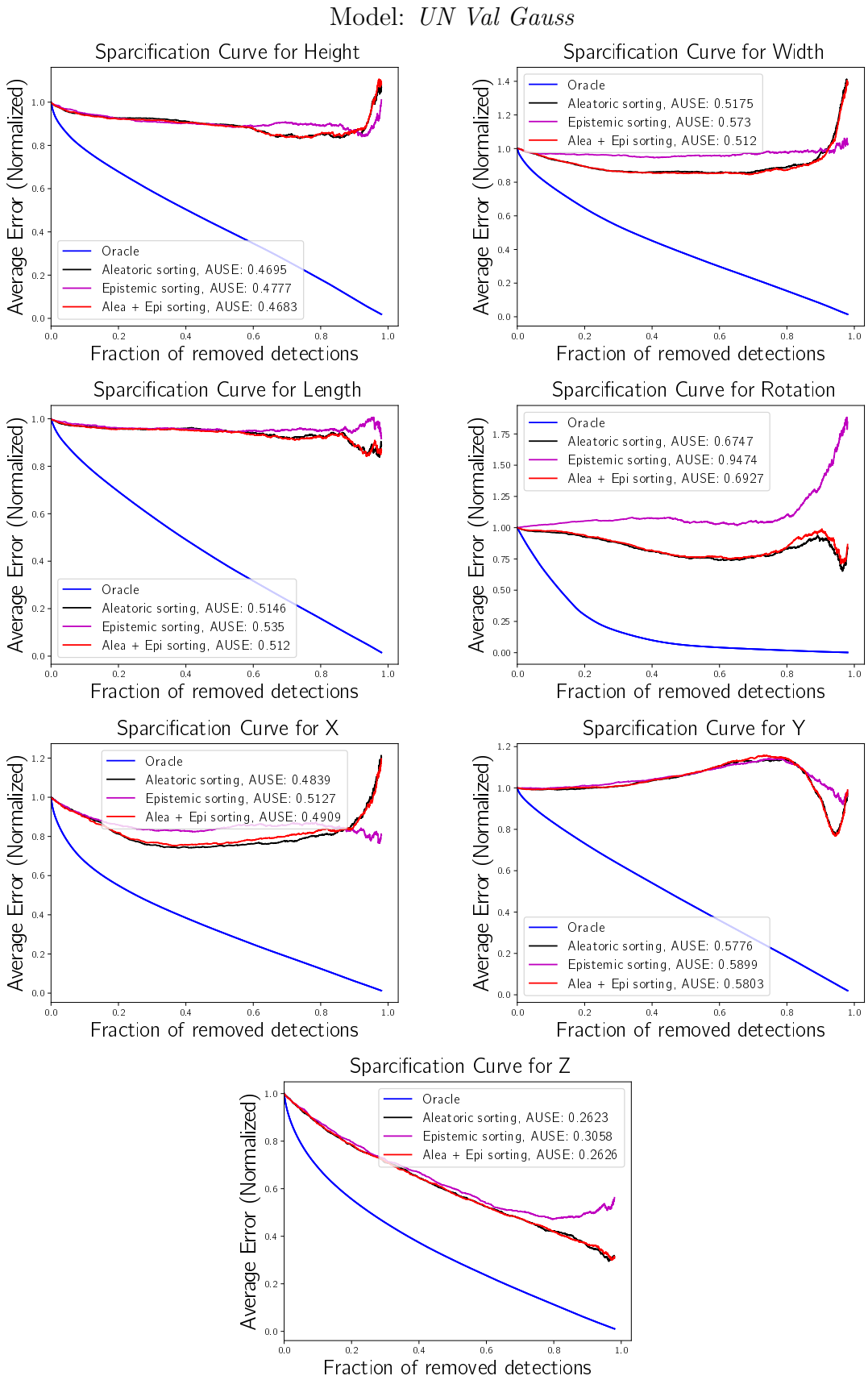


Figure A.7: Sparcification Curve plots of all the 3D variables for the model *UN Val Gauss*.

Model: *UN F1 Laplace*

2D IoU | BEV IoU | 3D IoU

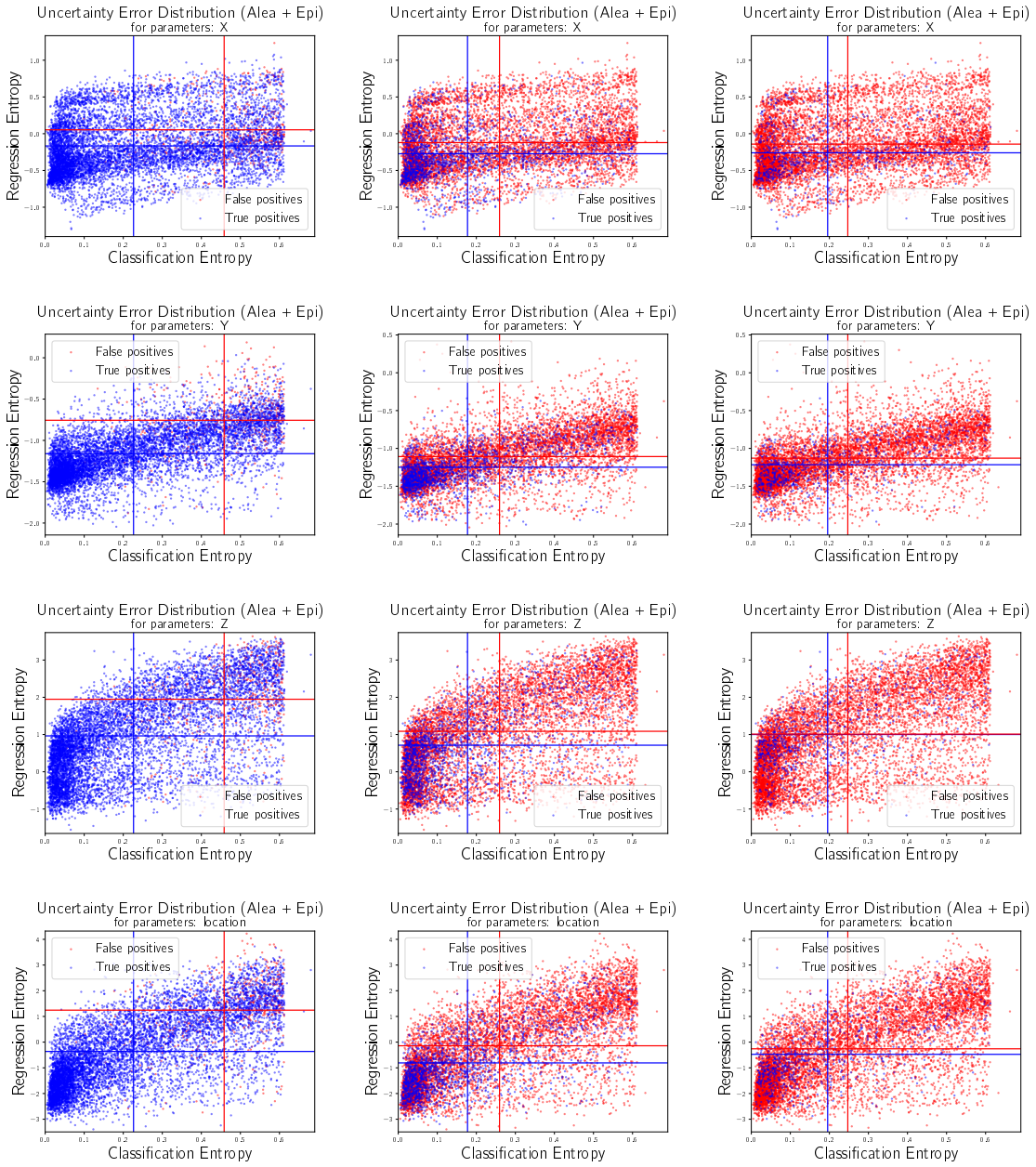


Figure A.8: Uncertainty Error plots for the 3D location and the (X, Y, Z) parameters separately with combined aleatoric and epistemic uncertainty. The model is *UN F1 Laplace* and the scoring functions are 2D, BEV and 3D, from left to right.

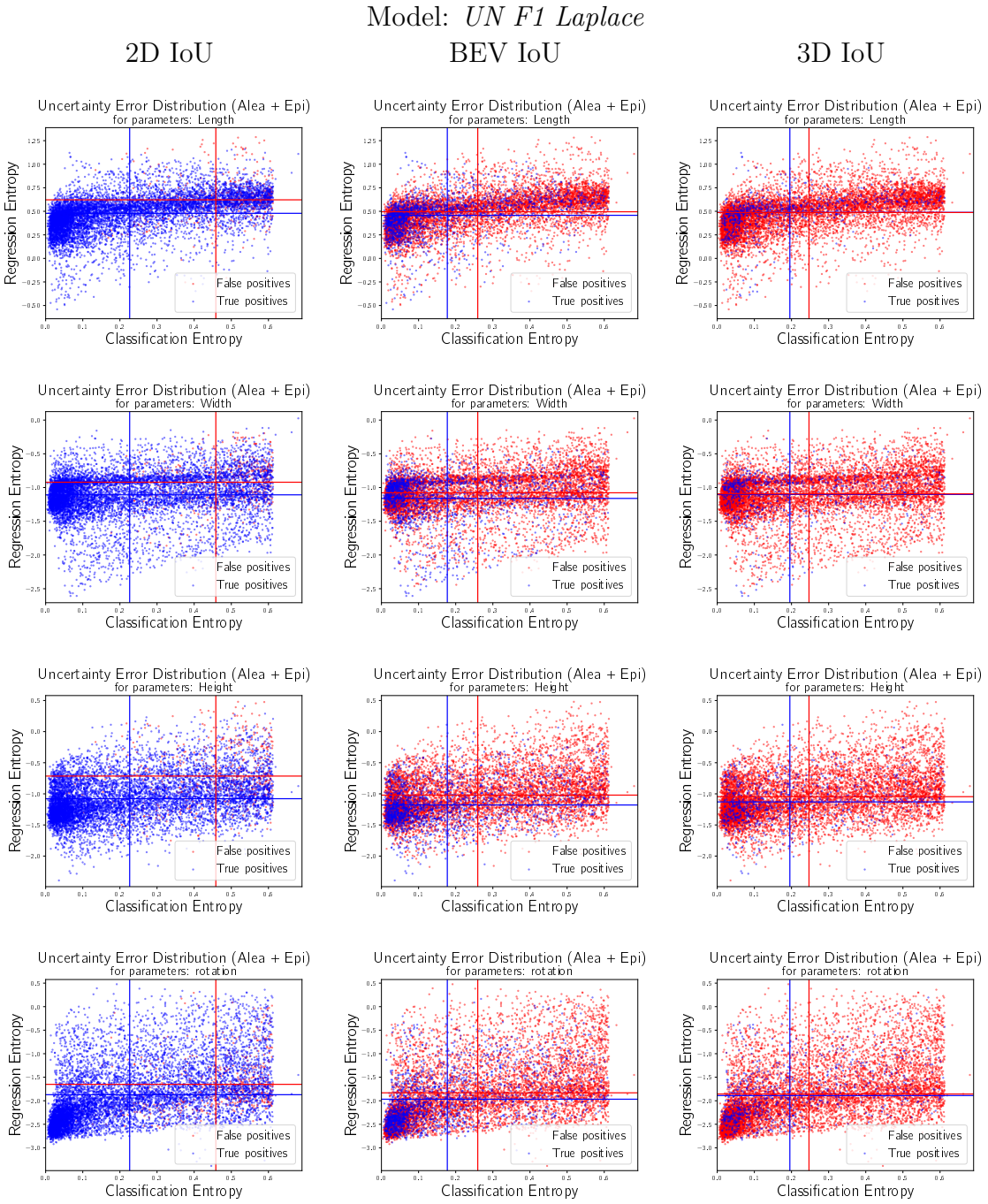


Figure A.9: Uncertainty Error plots for L, W, H, θ with combined aleatoric and epistemic uncertainty. The model is *UN F1 Laplace* and the scoring functions are 2D, BEV and 3D, from left to right.

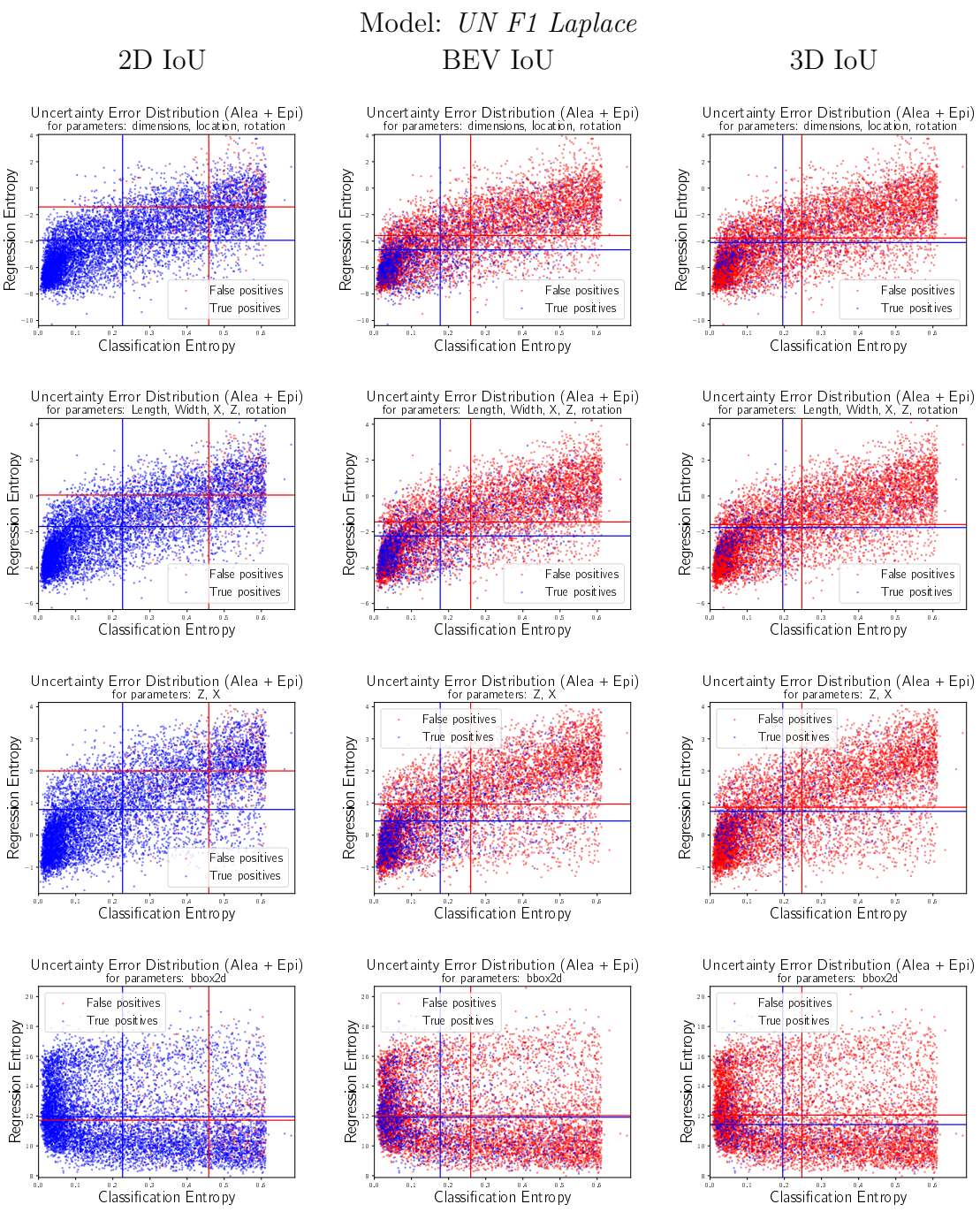


Figure A.10: Uncertainty Error plots for the combinations: $(H, W, L, X, Y, Z, \theta)$, (L, W, X, Z), (Z, X) and the 2D bounding box parameters. The uncertainty is combined aleatoric and epistemic uncertainty, the model is *UN F1 Laplace* and the scoring functions are 2D, BEV and 3D, from left to right.

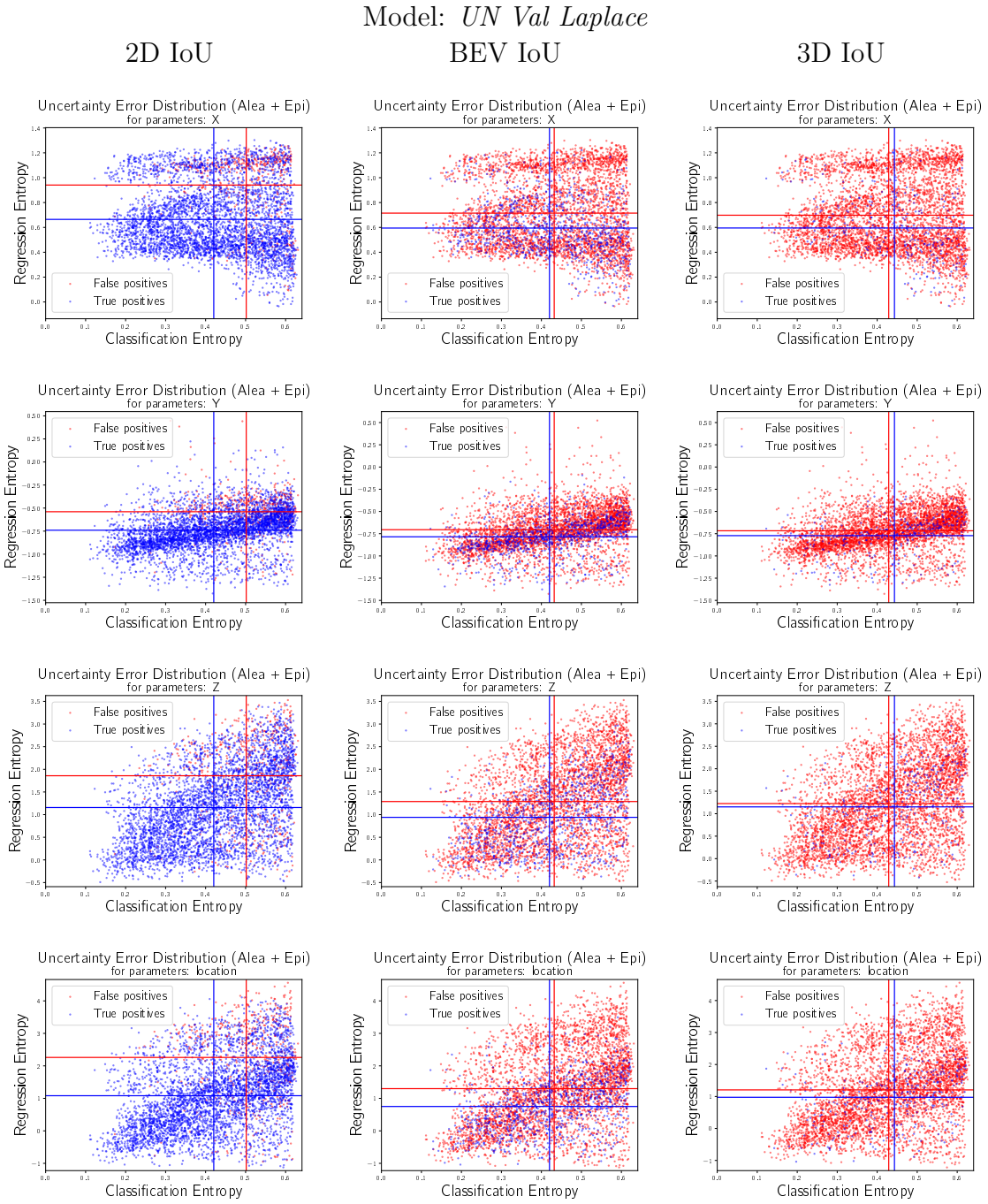


Figure A.11: Uncertainty Error plots for the 3D location and the (X, Y, Z) parameters separately with combined aleatoric and epistemic uncertainty. The model is *UN Val Laplace* and the scoring functions are 2D, BEV and 3D, from left to right.

Model: *UN Val Laplace*

2D IoU | BEV IoU | 3D IoU

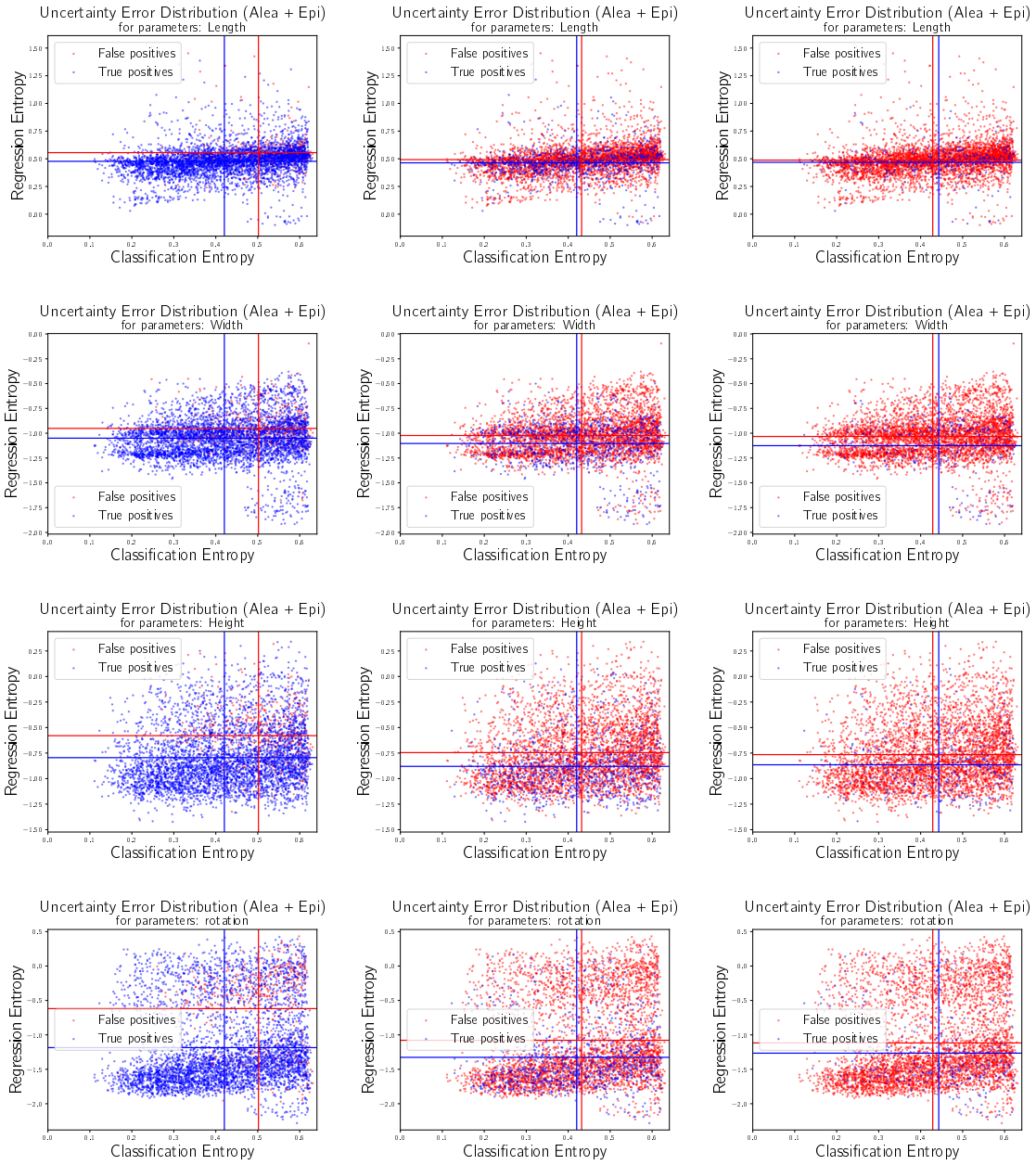


Figure A.12: Uncertainty Error plots for L, W, H, θ with combined aleatoric and epistemic uncertainty. The model is *UN Val Laplace* and the scoring functions are 2D, BEV and 3D, from left to right.

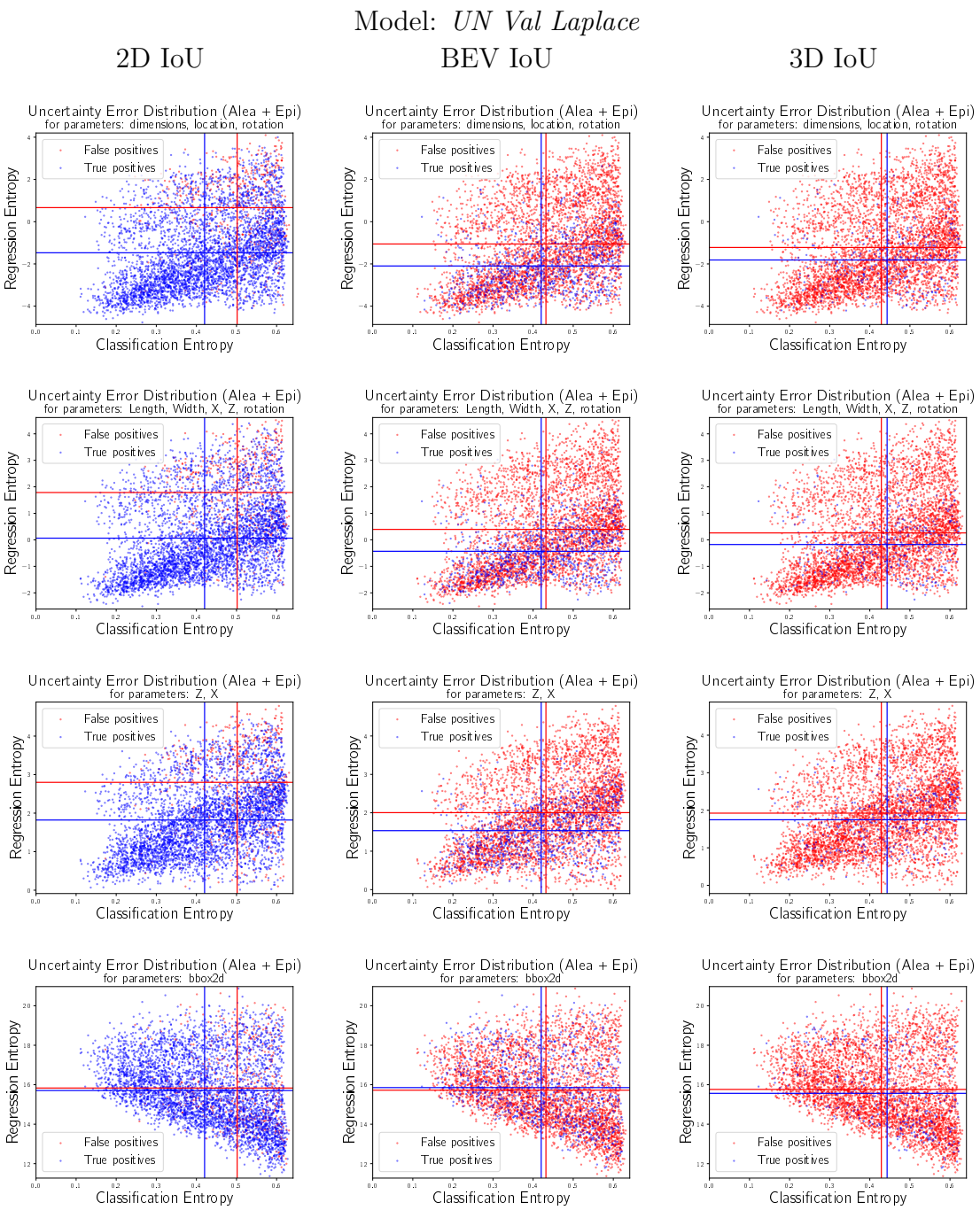


Figure A.13: Uncertainty Error plots for the combinations: $(H, W, L, X, Y, Z, \theta)$, (L, W, X, Z), (Z, X) and the 2D bounding box parameters. The uncertainty is combined aleatoric and epistemic uncertainty, the model is *UN Val Laplace* and the scoring functions are 2D, BEV and 3D, from left to right.

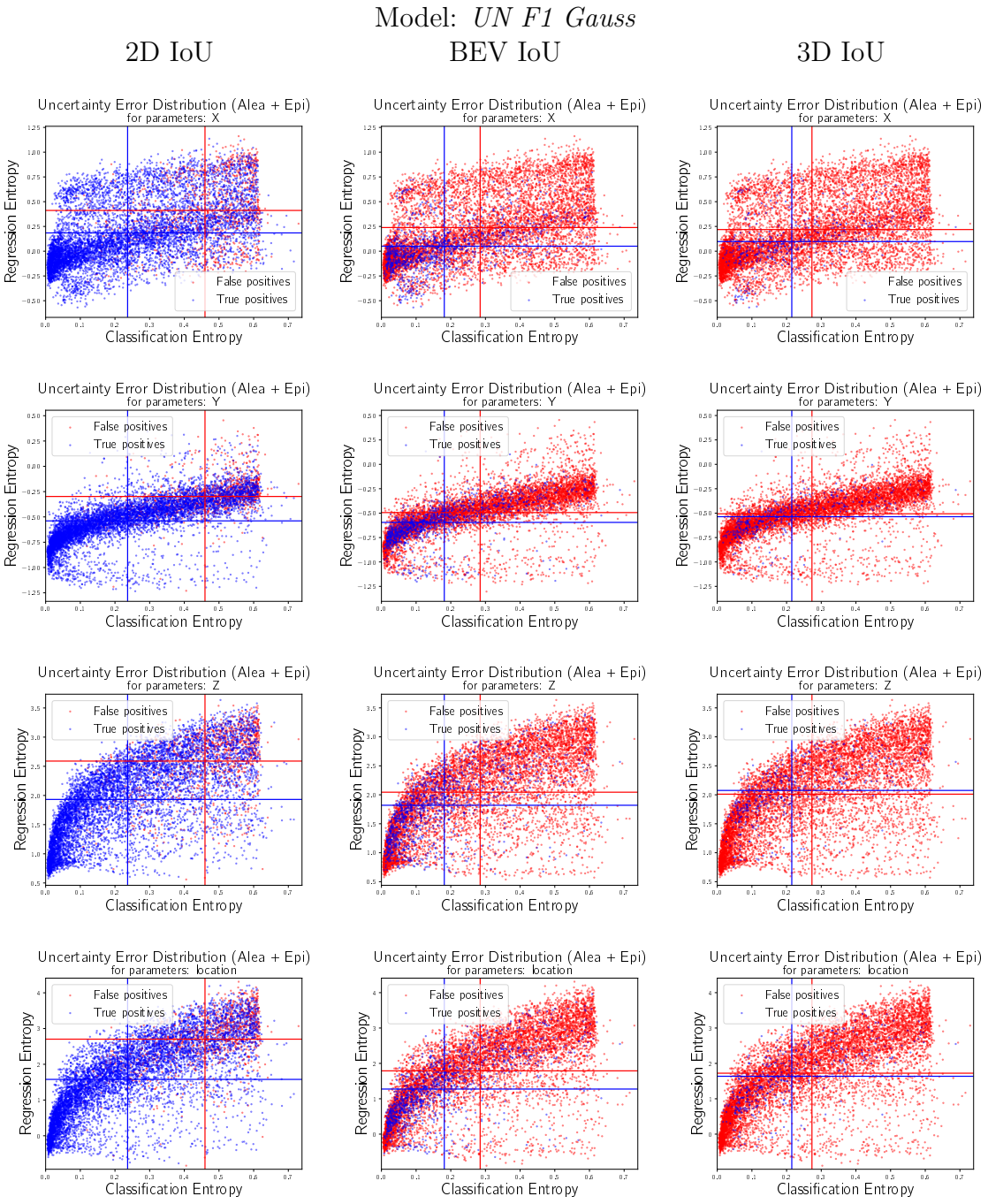


Figure A.14: Uncertainty Error plots for the 3D location and the (X, Y, Z) parameters separately with combined aleatoric and epistemic uncertainty. The model is *UN F1 Gauss* and the scoring functions are 2D, BEV and 3D, from left to right.

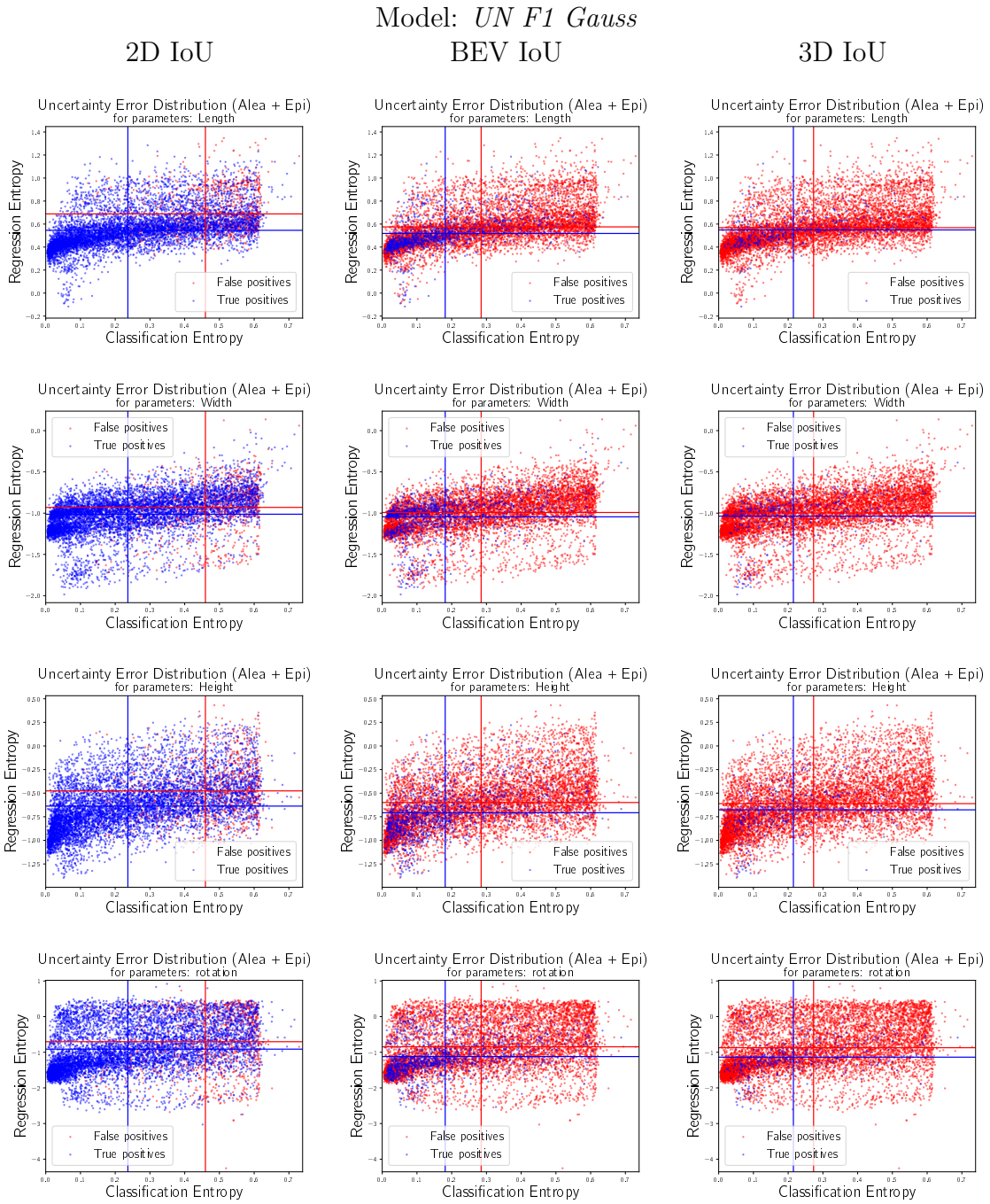


Figure A.15: Uncertainty Error plots for L, W, H, θ with combined aleatoric and epistemic uncertainty. The model is *UN F1 Gauss* and the scoring functions are 2D, BEV and 3D, from left to right.

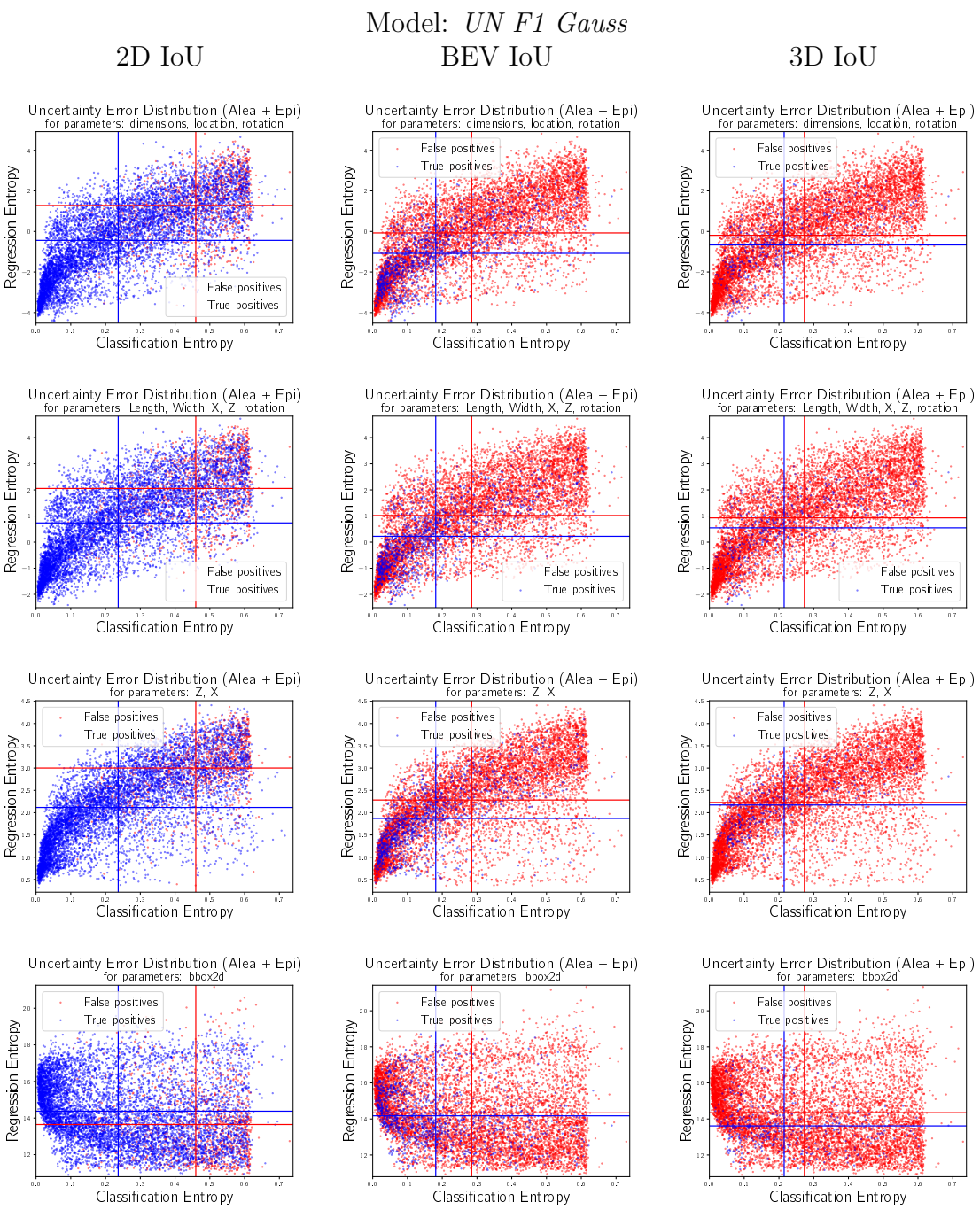


Figure A.16: Uncertainty Error plots for the combinations: $(H, W, L, X, Y, Z, \theta)$, (L, W, X, Z), (Z, X) and the 2D bounding box parameters. The uncertainty is combined aleatoric and epistemic uncertainty, the model is *UN F1 Gauss* and the scoring functions are 2D, BEV and 3D, from left to right.

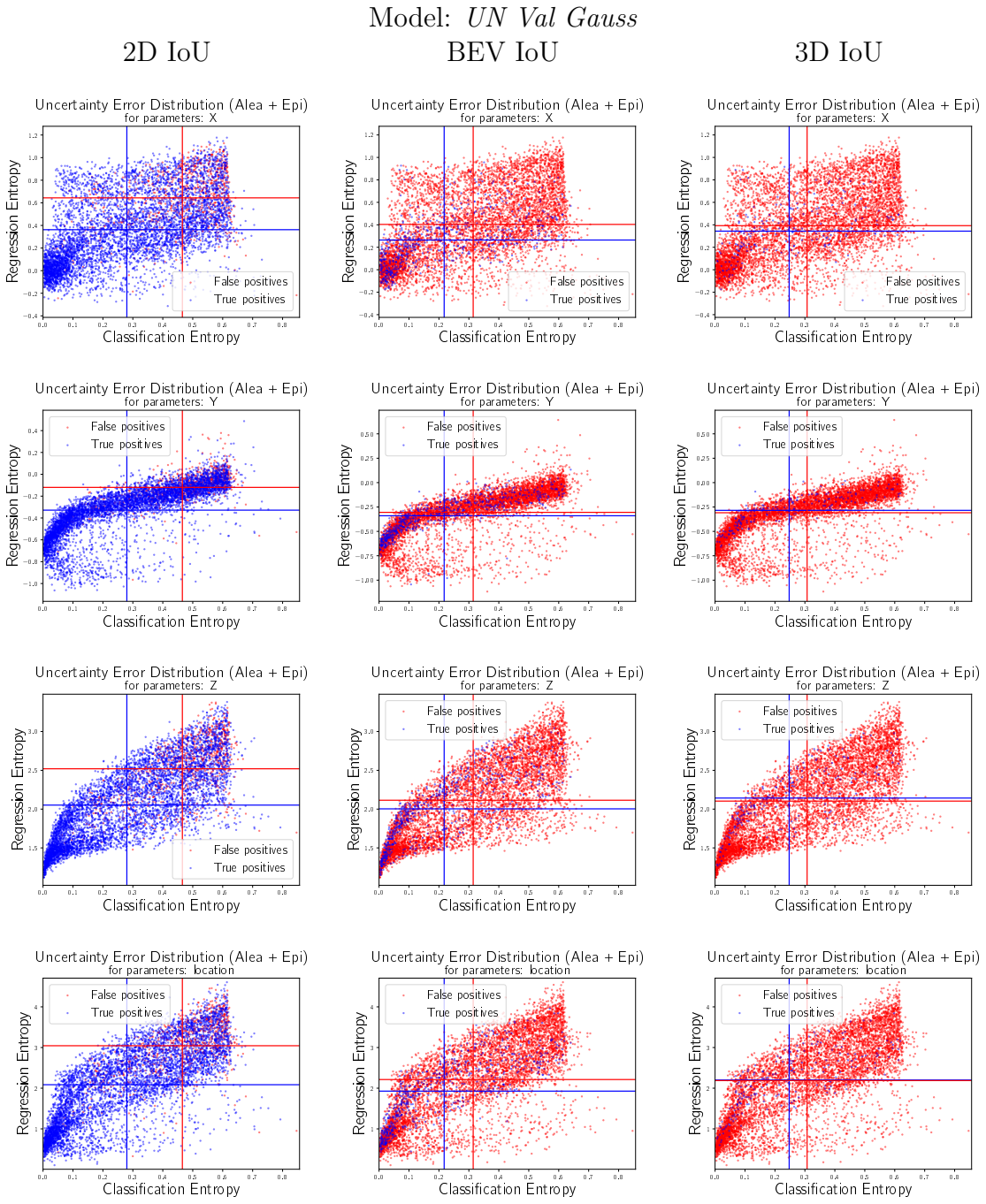


Figure A.17: Uncertainty Error plots for the 3D location and the (X, Y, Z) parameters separately with combined aleatoric and epistemic uncertainty. The model is *UN Val Gauss* and the scoring functions are 2D, BEV and 3D, from left to right.

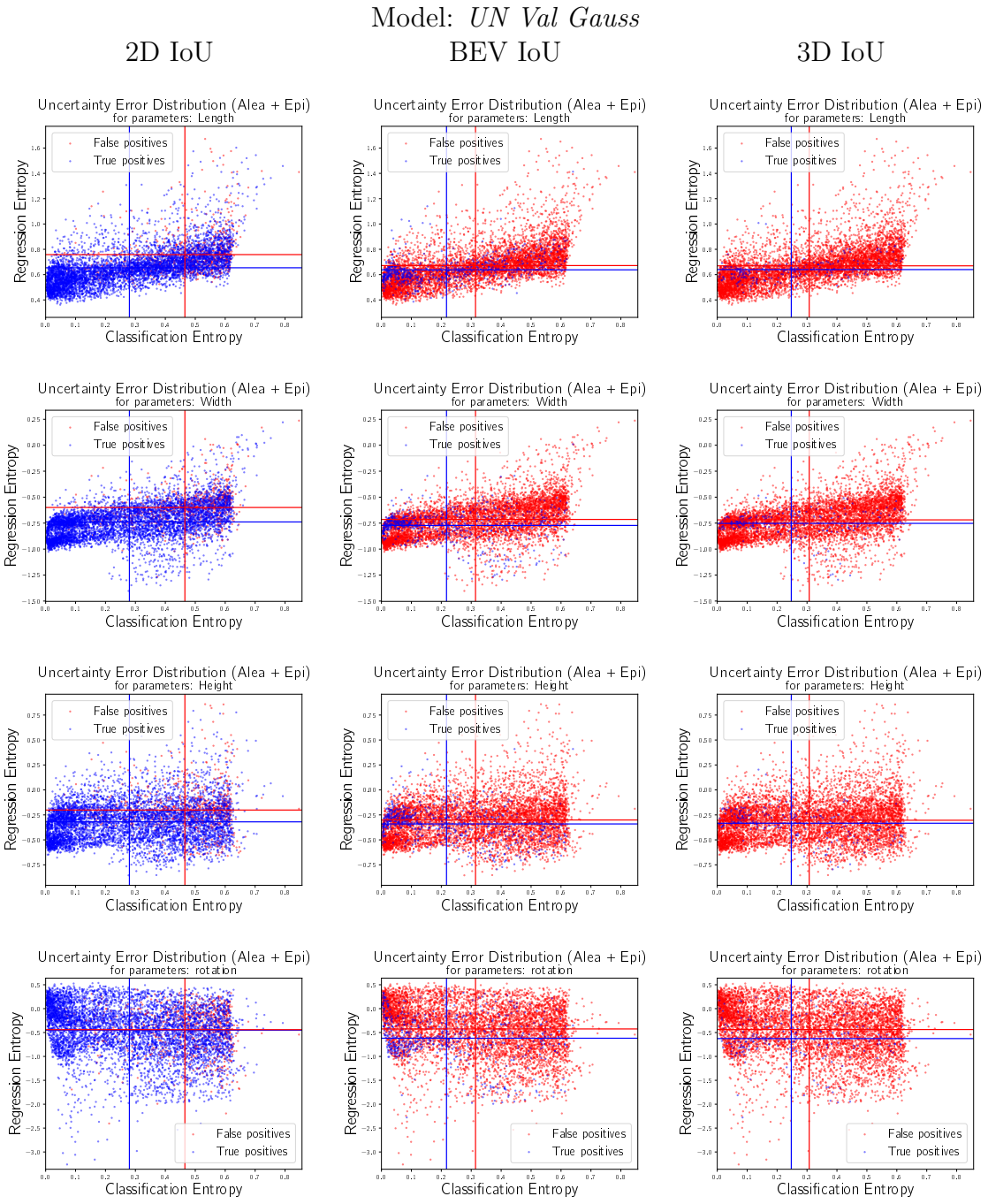


Figure A.18: Uncertainty Error plots for L, W, H, θ with combined aleatoric and epistemic uncertainty. The model is *UN Val Gauss* and the scoring functions are 2D, BEV and 3D, from left to right.

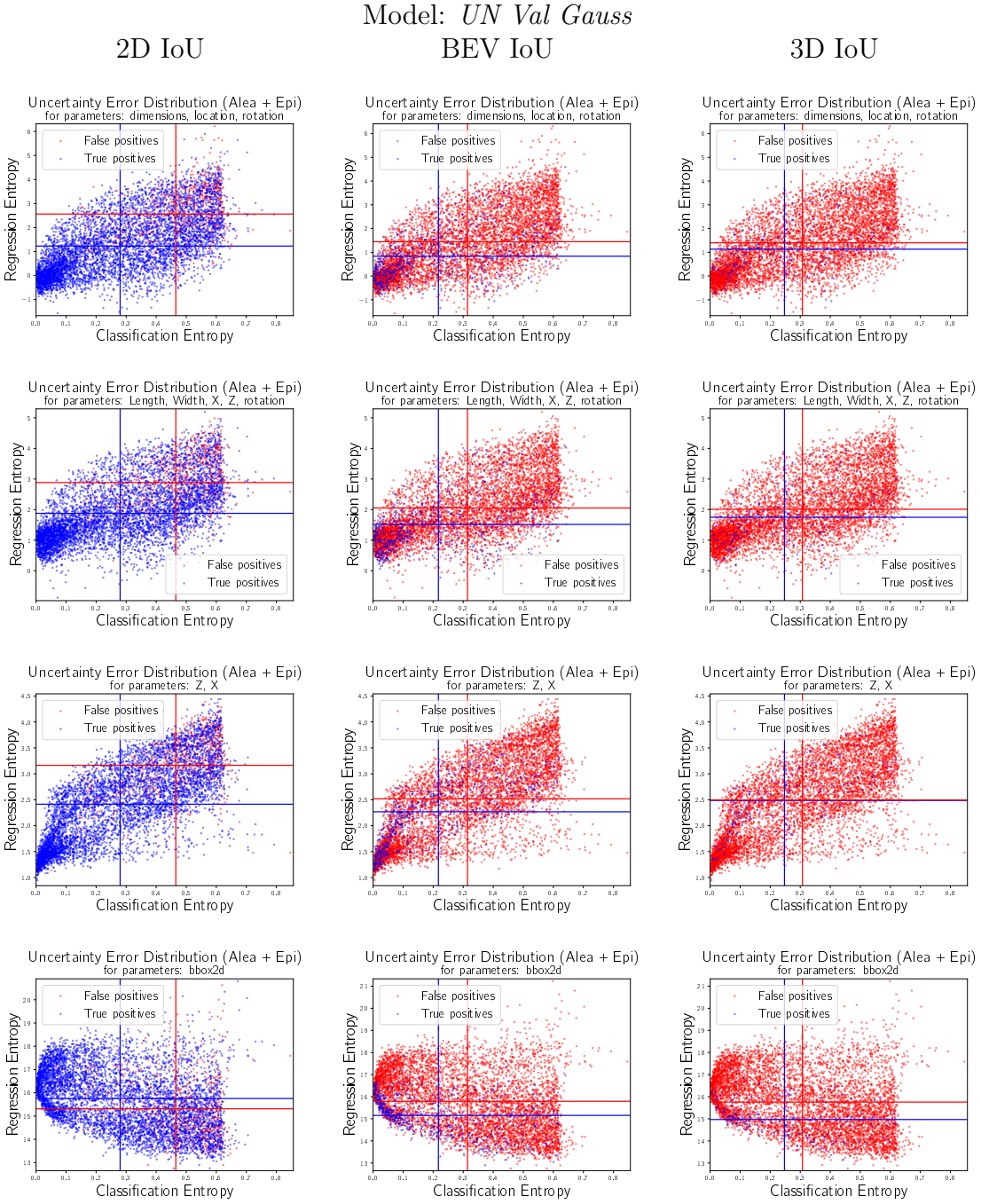


Figure A.19: Uncertainty Error plots for the combinations: $(H, W, L, X, Y, Z, \theta)$, (L, W, X, Z), (Z, X) and the 2D bounding box parameters. The uncertainty is combined aleatoric and epistemic uncertainty, the model is *UN Val Gauss* and the scoring functions are 2D, BEV and 3D, from left to right.