False Positive Detection and Prediction Quality Estimation for LiDAR Point Cloud Segmentation

1st Pascal Colling
Department of Mathematics
University of Wuppertal, Germany
pascal.colling@uni-wuppertal.de

2nd Matthias Rottmann
Department of Mathematics
University of Wuppertal, Germany
rottmann@uni-wuppertal.de

3rd Lutz Roese-Koerner
Aptiv, Wuppertal, Germany
lutz.roese-koerner@aptiv.de

4th Hanno Gottschalk
Department of Mathematics
University of Wuppertal, Germany
hanno.gottschalk@uni-wuppertal.de

***Abstract*—We present a novel post-processing tool for semantic segmentation of LiDAR point cloud data, called LidarMetaSeg, which estimates the prediction quality segmentwise. For this purpose we compute dispersion measures based on network probability outputs as well as feature measures based on point cloud input features and aggregate them on segment level. These aggregated measures are used to train a meta classification model to predict whether a predicted segment is a false positive or not and a meta regression model to predict the segmentwise intersection over union. Both models can then be applied to semantic segmentation inferences without knowing the ground truth. In our experiments we use different LiDAR segmentation models and datasets and analyze the power of our method. We show that our results outperform other standard approaches.**

***Index Terms*—deep learning, lidar point cloud, semantic segmentation, uncertainty quantification, automated driving**

I. Introduction

In the field of automated driving, scene understanding is essential. One possible solution for the semantic interpretation of scenes captured by multiple sensor modalities is LiDAR point cloud segmentation [1]–[4] (in the following LiDAR segmentation for brevity) where each point of the point cloud is assigned to a class of a given set. A segment is an area of points of the same class. Compared to camera images, a LiDAR point cloud is relatively sparse, but provides accurate depth information. Furthermore, since the LiDAR sensor in general is rotating, 360 degrees of the environment are considered. A summary for sensor modalities is given in [5]. In recent years, the performance of LiDAR segmentation networks has increased enormously [1]–[4], [6], but there are only few works on uncertainty quantification [2]. In applications of street scene understanding, safety and reliability of perception systems are just as important as their accuracy. To tackle this problem, we introduce a post-processing tool, called *LidarMetaSeg*, which estimates the segmentwise (i.e., per connected component of the predicted segmentation) prediction quality in terms of segmentwise intersection over union [7] (IoU) of the LiDAR segmentation model, see also fig. 1. This provides not only uncertainty quantification per predicted segment but also an online assessment of prediction quality.

H.G. and M.R. acknowledge financial support through the research consortium bergisch.smart.mobility funded by the ministry for economy, innovation, digitalization and energy (MWIDE) of the state North Rhine Westphalia under the grant-no. DMR-1-2.

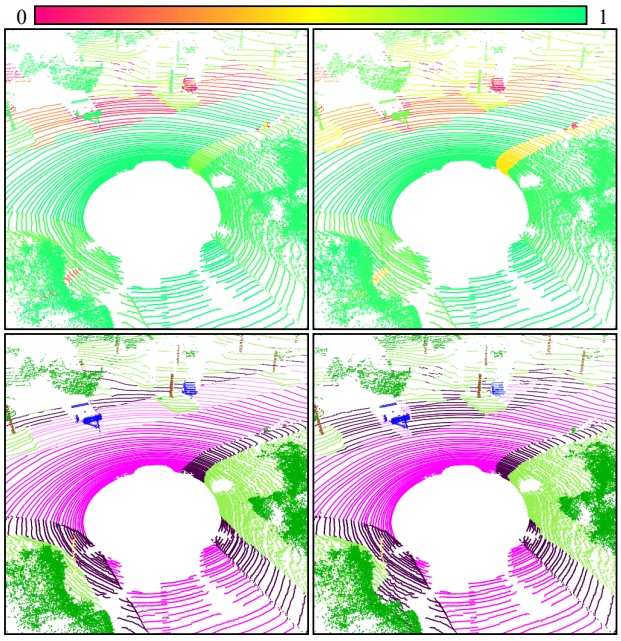


Fig. 1. A visualization of LidarMetaSeg containing the ground truth (bottom left), the LiDAR segmentation (bottom right), the LiDAR segmentation quality (top left) as the IoU of prediction and ground truth and its estimation obtained by LidarMetaSeg (top right). The higher the IoU, the better the prediction quality.

State-of-the-art LiDAR segmentation models are based on deep neural networks and can be grouped into two main approaches: projection-based (2D) and non-projection-based (3D) networks, cf. [8]. Projection-based networks like [1], [2], [9] use a spherical (2D) image representation of the point cloud. The predicted semantic categories on the image are thereafter reinserted along the spherical rays into the 3D point

cloud. This may contain some post-processing steps, like the k-nearest neighbor (kNN) approach, see [1]. Due to the representation of point clouds as projected images, the networks employed for LiDAR segmentation have architectures that often resemble image segmentation architectures. The non-projection-based networks, e.g. [3], [10], [11], process the point cloud directly in 3D space with or without different 3D representation approaches. For example, in [11], the network operates on the 3D point cloud without introducing an additional representation while in [3] the authors perform a 3D cylinder partition. A combination of a 2D and 3D representation of the point cloud is used in [4]. All current architectures, using a 2D or 3D representation or a combination of both provide the segmentation of the point cloud. Therefore, it is also possible to output the probabilities, which is the only prerequisite required for LidarMetaSeg.

Concerning uncertainty quantification in deep learning, Bayesian approaches like Monte Carlo (MC) dropout [12] are commonly used, e.g. in image-based object detection [13], image segmentation [14] and also in LiDAR object detection [15]. In object detection and instance segmentation, so called scores containing (un)certainty information are used, while this is not the case for semantic segmentation. The network SalsaNext [2] is for LiDAR segmentation and makes use of MC dropout to output the model (epistemic) and observation (aleatoric) uncertainty.

In our method LidarMetaSeg we first project the point cloud and the corresponding softmax probabilities of the network to a spherical 2D image representation, which are then used to compute different types of dispersion measures resulting in different dispersion heatmaps. To estimate uncertainty on segment level, we aggregate the dispersion measures with respect to each predicted segment. The IoU is commonly used to evaluate the performance of a segmentation model. For each predicted segment, we compute its IoU with the ground truth and call this *segmentwise* IoU. In our experiments we observe a strong correlation of the segmentwise IoU with the aggregated dispersion measures. Hence, we use the aggregated dispersion measures with additional information from the point cloud input to create a set of handcrafted features. The latter are used in post-processing manner as input for training *i)* a *meta classification model* to detect false positive segments, i.e., if the IoU is equal or greater than 0 and *ii)* a *meta regression model* to estimate the segmentwise IoU. Thus, we not only have a pointwise uncertainty quantification, given by the dispersion heatmaps, but also a false positive detection as well as a segmentation quality estimation on segment level.

The idea of meta classification and regression to detect false positives and to estimate the segmentwise prediction quality was first introduced in the field of semantic segmentation of images [16], called MetaSeg. The work presented in [17] goes in a similar direction, but for brain tumor segmentation. MetaSeg was further extended in other directions, i.e., for controlled false negative reduction [18], for time dynamic uncertainty estimates for video data [19], for taking resolution-dependent uncertainties into account [20] and as part of an active learning method [21]. Inspired by the possibility of representing the point could as a 2D image, our method LidarMetaSeg is an extension and further development of the original work. Therefore MetaSeg [16] is the most related work to our approach LidarMetaSeg, which up to now together with SalsaNext [2] are the only works in the direction of uncertainty quantification in LiDAR segmentation.

With MC dropout, SalsaNext follows a Bayesian approach to quantifying the model and the observation uncertainty. The uncertainty output is point-based and not segment-based, as in our approach. Also for MC dropout, the model has to infer one sample multiple times. LidarMetaSeg requires only a single network inference and estimates uncertainties by means of the network's class probabilities. In a 2D representation, these pixelwise uncertainty estimates can be viewed as uncertainty heatmaps. From those heatmaps, we compute aggregated uncertainties for each predicted segment, therefore clearly going beyond the stage of pixelwise uncertainty estimation. In contrast to MetaSeg for image segmentation, we not only use the network's output but also utilize information from the point cloud input, such as the intensity and range features provided for each point of the point cloud.

LidarMetaSeg is therefore a universal post-processing tool that allows for the detection of false positive segments as well as the estimation of segmentwise LiDAR segmentation quality. Besides that, the present work is the first one to provide uncertainty estimates on the level of predicted segments. We evaluate our method on two different datasets, SemanticKITTI [22] and nuScenes [23] and with three different network architectures, two projection-based models RangeNet++ [1], SalsaNext [2] and one non-projection-based model, Cylinder3D [3]. For meta classification, we achieve area under receiver operating characteristic curve (AUROC) and area under precision recall curve (AUPRC) [24] values of up to 91.16% and 74.35%, respectively. For the meta regression, we achieve coefficient of determination R^2 values of up to 66.69%. We show that our aggregated measures – in terms of meta classification and regression – lead to a significant performance gain in comparison to when only considering a single uncertainty metric like the segmentwise entropy.

II. Method

LidarMetaSeg is a post-processing method for LiDAR semantic segmentation to estimate the segmentwise prediction quality. It consists of a meta classification and a meta regression model that for each predicted segment classifies whether it has an IoU equal to or greater than 0 with the ground truth and predicts the segmentwise IoU with the ground truth, respectively. The method works as follows: in a preprocessing step we project each sample, i.e., the point cloud, the corresponding network probabilities and the labels into a spherical 2D image representation. In a next step and based on the projected data, we compute dispersion measures and other features like it is done for image data in [16], [18], [20]. Afterwards we identify the segments of a given semantic class and aggregate the pixelwise values from the previous step

on a segment level. In addition, we compute the IoU of each predicted segment with the ground truth of the same class. This results in a structured dataset, which consist of the coefficients of the aggregated dispersion measures as well as additional features and of the target variable – the $IoU \in [0, 1]$ for the task of meta regression or the binary variable $IoU = 0, > 0$ ($IoU = 0$ as indicator for a false positive) for the task of meta classification – for each segment. We fit a classification and a regression model to this dataset. In the end, we re-project the meta classification and regression from the image representation to the point cloud.

A. Preprocessing

A sample of input data for LidarMetaSeg is assumed to be given on point cloud level and contains the following:

- *point cloud* $\tilde{p} \in \mathbb{R}^{m\times 4}$, $\tilde{p}_j = (x_j, y_j, z_j, i_j)$ with $x_j, y_j, z_j \in \mathbb{R}$ Cartesian coordinates and intensity $i_j \in \mathbb{R}_+$ for $j = 1, \dots, m$ with m the number of points in the LiDAR point cloud,
- *ground truth / labels* $\tilde{y}^* \in \mathcal{C}^m$ with $\mathcal{C} = \{1, \dots, n\}$ the set of n given classes,
- *probabilities* $\tilde{y}^{prob} = f(\tilde{p}) \in \mathbb{R}^{m\times n}_{[0,1]}$ of a LiDAR segmentation network f, given as softmax probabilities,
- *prediction* $\tilde{y} = \arg\max_{c\in\mathcal{C}} \tilde{y}^{prob}$.

Typically, one is also interested in the range of a given point in the point cloud, which is part of most LiDAR segmentation networks' input. Since the ego car is located in the origin of the coordinate system, this quantity is given by $r_j = \sqrt{x_j^2 + y_j^2 + z_j^2}$ for each $\tilde{p}_j$.

The projection of a point cloud to a spherical 2D image representation follows two steps: a transformation from Cartesian to spherical coordinates and then a transformation from spherical to image coordinates. The spherical coordinates are given as $(r_j, \theta_j, \varphi_j)$ with range r_j, polar angle θ_j and azimuth angle φ_j. The transformation for the Cartesian to spherical coordinates is given by

$$\theta_j = \arcsin\left(\frac{z_j}{r_j}\right) \quad \text{and} \quad \varphi_j = \arctan\left(\frac{y_j}{x_j}\right) \tag{1}$$

and r_j for $j = 1, \dots, m$. Based on the spherical coordinates we get the image coordinates (u, v) with the equation

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - (\varphi_j \pi^{-1})\right] w \\ \left[1 - (\theta_j + f_{ver^+}) f_{ver}^{-1}\right] h \end{pmatrix} \tag{2}$$

with (w, h) the width and height of the image and $f_{ver} = f_{ver^+} + f_{ver^-}$ the vertical field of view (FOV) of the LiDAR sensor.

In order to get an image representation where each point correspond to one pixel and vice versa, we need the number of channels, the angular resolution and the horizontal FOV of the LiDAR sensor. To this end, we define the height h as the number of channels and the width w as the quotient of the horizontal FOV f_{hor} (which is in general 360, since the LiDAR sensor is rotating) and the angular resolution α, i.e.,

$$w = \frac{f_{hor}}{\alpha}. \tag{3}$$

Thus, the image representation – using the explicit sensor information – has as many entries or pixels as the point cloud can have as maximum number of points. Unfortunately there are still some technical reasons, due to which it can happen that multiple points are projected to the same pixel, e.g. ego-motion compensation or overlapping channel angles. More details concerning such projection errors can be found in [25]. With the projection proposed above this happens rarely enough, so that this event is negligible.

Following the projection above, we denote the projected 2D representation similar as before, but without $\tilde{\cdot}$, i.e.,

- image representation (of the point cloud $\tilde{p}$) $F = (F^x, F^y, F^z, F^i, F^r)$, $F \in \mathbb{R}^{w\times h\times 5}$,
- ground truth / labels $y^* \in \mathcal{C}^{w\times h}$,
- probabilities $y^{prob} \in \mathcal{C}^{w\times h\times n}$,
- prediction $y \in \mathcal{C}^{w\times h}$.

The proposed image projection yields a sparse image representation. However, our post-processing approach LidarMetaSeg is based on connected components of the segmentation. In order to identify connected components (segments) of pixels in the 2D image resulting from the projection, we fill these gaps by setting any empty entry $l := (u, v)$ (entries without a corresponding point in the point cloud) to a value of one of its nearest neighbors that received a value from the projection. An example of such a filled image representation is shown in fig. 2, left panel. In the following we only consider the filled image representations. We store the information which pixel received its value via projection and which one via fill-in in a binary mask of width w and height h denoted by δ where 1 represents a projected point and 0 a filled entry, i.e.,

$$\delta_l = \begin{cases} 1, & l \text{ corresponds to a projected point,} \\ 0, & \text{else.} \end{cases} \tag{4}$$

For simplicity, we refer the filled image representations F (that are input quantities for the segmentation network) as feature measures.

B. Dispersion Measures and Segmentwise Aggregation

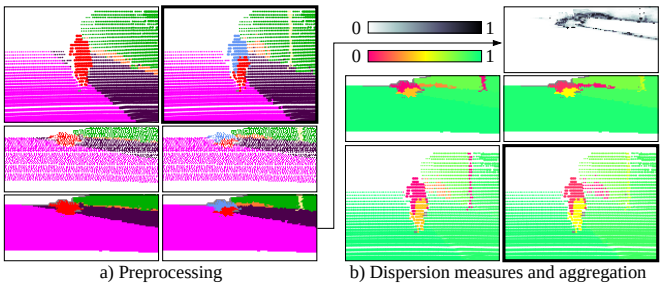
First we define the dispersion and feature measures and afterwards the segmentwise aggregation.

a) Dispersion and Feature Measures: Based on the probabilities $y^{prob} \in [0, 1]^{w\times h\times n}$, we define the dispersion measures *entropy* E_l, *probability difference* D_l and *variation ratio* V_l at pixel position $l = (u, v)$ as follows:

$$E_l = -\frac{1}{\log(n)} \sum_{c=1}^{n} y_{l,c}^{prob} \log\left(y_{l,c}^{prob}\right), \tag{5}$$

$$D_l = 1 - \max_{c_1\in\mathcal{C}} y_{l,c_1}^{prob} + \max_{c_2\in\mathcal{C}\setminus c_1} y_{l,c_2}^{prob}, \tag{6}$$

$$V_l = 1 - \max_{c\in\mathcal{C}} y_{l,c}^{prob}. \tag{7}$$



a) Preprocessing b) Dispersion measures and aggregation

Fig. 2. Visual examples of our method LidarMetaSeg. The left panel shows the preprocessing part: the ground truth (top left) and the prediction (top right) of the point cloud as well as the corresponding sparse (middle) and filled (bottom) image representations. The right panel visualizes a dispersion heatmap, the segmentwise prediction quality and its estimation: the probability difference heatmap of the prediction-based probabilities (top right), where higher values correspond to higher uncertainty, in the middle the true (left) and estimated (right) IoU_{adj} values for the image representation and in the bottom part the corresponding visualizations after the re-projection to the point cloud. The prediction of the point cloud and the corresponding prediction quality estimation is highlighted.

In addition, the feature measures *coordinates*, *intensity* and *range* at position l are given by the image representation

$$F_l^\sharp,\ \sharp \in \{x, y, z, i, r\}. \tag{8}$$

For the sake of brevity, we define the set of dispersion and features measures

$$\mathcal{M} = \{E, D, V, F^x, F^y, F^z, F^i, F^r\} \tag{9}$$

omitting the index for the position l as this will follow from the context. Note that, due to the position dependence, each element of $\mathcal{M}$ can be considered as a heatmap.

b) Segmentwise Aggregation: For a given prediction $y \in \mathcal{C}^{w\times h},\ \mathcal{C} = \{1, \dots, n\}$ and the corresponding ground truth $y^* \in \mathcal{C}^{w\times h},\ \mathcal{C} = \{1, \dots, n\}$, we denote $\mathcal{K}_y$ and $\mathcal{K}_{y^*}$ the set of connected components (segments) in the prediction and the ground truth, respectively. A connected component k is a set of pixels that are adjacent to each other and belong to the same class, see also fig. 2, left panel. For each segment $k \in \mathcal{K}_y$, we define the following quantities. Additionally and in order to count only the pixels with a corresponding point in the point cloud, we introduce the restriction by the corresponding binary mask δ with $|\cdot|_\delta$.

- The interior $k_{in} = \{(u, v) \in k : [u\pm1]\times[v\pm1] \in k\} \subset k$, i.e., a pixel $l = (u, v)$ is an element of k_{in} if all eight neighboring pixels are an element of k,
- the boundary $k_{bd} = k \setminus k_{in}$,
- the pixel sizes $S = |k|,\ S_{in} = |k_{in}|,\ S_{bd} = |k_{bd}|$,
- the segment size in the point cloud $SP = |k|_\delta$.

Furthermore, we define the target variables IoU and the so called adjusted IoU_{adj} as follow:

- IoU: let $\mathcal{K}_{y^*}|_k$ be the set of all $k' \in \mathcal{K}_{y^*}$ that have non-trivial intersection with k and whose class label equals the predicted class for k, then

$$IoU(k) = \frac{|k \cap K'|_\delta}{|k \cup K'|_\delta}, \qquad K' = \bigcup_{k' \in \mathcal{K}_{y^*}|_k} k', \tag{10}$$

- the adjusted IoU_{adj} does not count pixels in the ground truth segment that are not contained in the predicted segments, but in other predicted segments of the same class: let $Q = \{q \in \mathcal{K}_y : q \cap K' \neq \emptyset\}$, then

$$IoU_{\mathrm{adj}}(k) = \frac{|k \cap K'|_\delta}{|k \cup (K' \setminus Q)|_\delta}. \tag{11}$$

In cases where a ground truth segment is covered by more than one predicted segment of the same class, each predicted segment would have a low IoU, while the predicted segments represent the ground truth quite well. As a remedy, the adjusted IoU_{adj} was introduced in [16] to not punish this situation. The adjusted IoU_{adj} is more suitable for the task of meta regression. For the meta classification it holds $IoU = 0, > 0 \Leftrightarrow IoU_{\mathrm{adj}} = 0, > 0$.

Based on the previous definitions, we define the dispersion and feature measures:

- the mean μ and variance υ metrics

$$\mu M_\sharp := \mu(M_\sharp) = \frac{1}{S_\sharp} \sum_{l \in k_\sharp} M_l \tag{12}$$

$$\upsilon M_\sharp := \upsilon(M_\sharp) = \mu(M_\sharp^2) - \mu(M_\sharp)^2 \tag{13}$$

  for $\sharp \in \{\_, in, bd\}$ and $M \in \mathcal{M}$,
- the relative sizes $\bar{S} = S/S_{bd},\ \bar{S}_{in} = S_{in}/S_{bd}$,
- the relative mean and variance metrics

$$\tau\bar{M} = \tau M \bar{S} \tag{14}$$

$$\tau\bar{M}_{in} = \tau M \bar{S}_{in} \tag{15}$$

  for $\tau \in \{\mu, \upsilon\}$ and $M \in \mathcal{M}$,
- the ratio of the neighborhood's correct predictions of each class

$$N_c = \frac{1}{|k_{nb}|} \sum_{l \in k_{bd}} \mathbb{1}_{\{c=y_l\}} \qquad \forall c \in \mathcal{C} \tag{16}$$

  with k_{nb} the set of k neighbors, i.e., $k_{nb} = \{l' \in [u \pm 1] \times [v \pm 1] \subset w \times h : (u, v) \in k, l' \notin k\}$,
- the mean class probabilities

$$P_c = \frac{1}{S} \sum_{l \in k} y_{l,c}^{prob} \qquad \forall c \in \mathcal{C}. \tag{17}$$

Typically, the dispersion measures E_l, D_l, V_l are large for $l \in k_{bd}$. This motivates the separate treatment of interior and boundary measures. Furthermore we observe a correlation between fractal segment shapes and a bad or wrong prediction, which motivates the relative sizes $\bar{S},\ \bar{S}_{in}$. In summary, we have $86 + 2n$ metrics: the (relative) mean and variance metrics $\tau M, \tau M_{in}, \tau M_{bd}, \tau\bar{M}, \tau\bar{M}_{in}\ \forall \tau \in \{\mu, \upsilon\},\ \forall M \in \mathcal{M}$, the (relative) size metrics $S, S_{in}, S_{bd}, \bar{S}, \bar{S}_{in}, SP$ as well as $N_c, P_c\ \forall c \in \mathcal{C}$. An example of the pixelwise dispersion measures as well as the segmentwise IoU_{adj} values and its prediction is shown in fig. 2, right panel.

With the exception of the segmentwise IoU and IoU_{adj} values, all quantities defined above can be computed without the knowledge of the ground truth.

III. Numerical Experiments

For numerical experiments we used two datasets: SemanticKITTI [22] and nuScenes [23]. For meta classification and regression we deploy XGBoost [26]. Other classification and regression methods like linear / logistic regression, neural networks of tree based ensemble methods [27] are also possible. However, as shown in [19], XGBoost leads to the best results. Due to the reason mentioned in the previous section, the target variable for the meta regression (and classification) is the adjusted IoU_{adj}.

First, we describe the settings of the experiments for both datasets and evaluate the results for the false positive detection and the segmentwise prediction quality estimation when using all metrics presented in the previous section. Afterwards we conduct an analysis of the metrics and the meta classification model.

A. SemanticKITTI

The SemanticKITTI dataset [22] contains street scenes from and around Karlsruhe, Germany. It provides 11 sequences with about 23K samples for training and validation, consisting of 19 classes. The data is recorded with a Velodyne HDL-64E LiDAR sensor, which has 64 channels and a (horizontal) angular resolution of 0.08°. Furthermore the data is recorded and annotated with 10 frames per second (fps) and each point cloud contains about 120K points. The authors of the dataset recommend to use all sequences to train the LiDAR segmentation model, except sequence 08, which should be used for validation.

For the experiments we used three pretrained LiDAR segmentation models, two projection-based models, i.e., RangeNet++ [1] and SalsaNext [2], and one non-projection-based model, i.e., Cylinder3D [3], which followed the recommended data split. For RangeNet++ and SalsaNext, the softmax probabilities are given for the 2D image representation prediction. As we assume that softmax probabilities are given for the point cloud, we consider this representation as the starting point and re-project the softmax probabilities to the point cloud.

After the re-projection from the 2D image representation prediction to the point cloud, both models have an additional kNN post-processing step to clean the point cloud from undesired discretization and inference artifacts [1], which may results in changing the semantic class of a few points. To take this post-processing step into account, we set the values of the cleaned points in the corresponding softmax probabilities of the point cloud to 1 and all other values to 0. Therefore the softmax condition (the sum of all probability values of a point is equal to 1 and all values are between 0 and 1) is met and the adjusted prediction is equal to the argmax of the probabilities. We do not expect other approaches to significantly change the results since we aggregate our dispersion measures and the number of modified points is small.

Following our method, the image representation of the point cloud data is of size $(w, h) = (4500, 64)$, cf. (3).

Most deep learning models tend to overfit. Therefore we only use samples for LidarMetaSeg, which are not part of the training data of the segmentation network, as overfitted models affects the dispersion measures. Thus, we only use sequence 08 for our experiments. Computing the connected components and metrics yields approx. 3.4M segments for each network. Most of the segments are very small. Therefore we follow a similar segment exclusion rule as in MetaSeg [16], where segments with empty interior, $S_{in} = 0$, are excluded. Here, we exclude segments consisting of less than 10 LiDAR points, i.e., $SP < 10$, also shown in gray color in fig. 2. Hence, we reduce the number of segments to approx. 0.45M but we retain 99% of the data measured in terms of the number of points. We tested the dependence of our results under variation of the exclusion size SP. The results were very similar to the results we present in the following.

For training and validation of LidarMetaSeg we split sequence 08 and the corresponding connected components and metrics into 10 disjoint sub-sequences. These sub-sequences are used for a 10-fold cross validation. A cross validation over all samples would yield highly correlated training and validation splits as all sequences are recorded with 10 fps. The results for the meta classification and regression are given in table I. For all three models we achieve a validation accuracy between 85.50% and 88.37%, see row 'ACC LMS' (short for LidarMetaSeg). The accuracy of random guessing ('ACC naive baseline') is between 78.42% and 84.53% which directly amounts to percentage of segments with an $IoU_{\mathrm{adj}} > 0$.

For each method, the accuracy values correspond to a single decision threshold. In contrast to that, the AUROC and AUPRC are obtained by varying the decision threshold of the classification output. The AUROC essentially measures the overlap of distributions corresponding to negative and positive samples; this score does not place more emphasis on one class over the other in case of class imbalance. The ACC of random guessing indicates the class imbalance: about 80% of the segments have an $IoU_{\mathrm{adj}} > 0$ and 20% of the segments have an $IoU_{\mathrm{adj}} = 0$, i.e., they are false positives. The underlying precision recall curve of the AUPRC ignores true negatives and emphasizes the detection of the positive class (false positives).

Using the metrics of the previous section (LMS) for the meta classification yields AUROC values above 90% and AUPRC up to 74.35%. For the meta regression we achieve R^2 values between $61.57 - 66.69\%$. Fig. 3 depicts the quality of predicting the IoU_{adj}. A visualization of estimating the IoU_{adj} is shown in fig. 2 and in the supplementary video[1].

B. NuScenes

The nuScenes dataset [23] contains street scenes from two cities, Boston (US) and Singapore. It provides 700 sequences for training and 150 sequences for validation. Each sequence contains about 40 samples which amounts to a total of 34K key frames. The dataset has 16 classes and is recorded and annotated with 2 fps. The LiDAR sensor has 32 channels and

[1] https://youtu.be/907jJSRgHUk

TABLE I

RESULTS FOR META CLASSIFICATION AND REGRESSION, AVERAGED OVER 10 RUNS. THE NUMBERS IN THE BRACKETS DENOTE STANDARD DEVIATIONS OF THE COMPUTED MEAN VALUES. THE BEST RESULTS IN TERMS OF ACC, AUROC, AUPRC AND R^2 ON THE VALIDATION DATA ARE HIGHLIGHTED.

		SemanticKITTI						nuScenes	
		RangeNet++		SalsaNext		Cylinder3D		Cylinder3D	
		training	validation	training	validation	training	validation	training	validation
Classification $IoU_{\text{adj}} = 0, > 0$									
ACC	LMS	92.26%(±0.25%)	**85.50**%(±3.12%)	92.43%(±0.21%)	**86.26**%(±2.75%)	92.97%(±0.21%)	**88.37**%(±2.89%)	92.96%(±0.08%)	**91.00**%(±0.60%)
	LMS w/o features	90.64%(±0.30%)	85.10%(±3.21%)	90.77%(±0.23%)	86.02%(±2.86%)	91.68%(±0.27%)	88.22%(±2.93%)	92.31%(±0.13%)	90.62%(±1.16%)
	Entropy	79.37%(±0.37%)	79.21%(±3.17%)	80.14%(±0.33%)	80.06%(±2.75%)	85.35%(±0.30%)	85.24%(±2.78%)	89.91%(±0.14%)	89.86%(±1.17%)
	LMS ∪ MCDO			92.48%(±0.20%)	86.26%(±2.78%)				
	naive baseline	78.42%		79.36%		84.53%		89.85%	
AUROC	LMS	97.19%(±0.12%)	**90.58**%(±1.89%)	97.22%(±0.10%)	91.16%(±1.75%)	96.86%(±0.10%)	**90.78**%(±2.36%)	94.81%(±0.10%)	**90.05**%(±1.11%)
	LMS w/o features	95.97%(±0.13%)	89.85%(±1.96%)	95.95%(±0.10%)	90.81%(±1.80%)	95.52%(±0.18%)	90.57%(±2.46%)	93.47%(±0.10%)	89.09%(±1.29%)
	Entropy	81.45%(±0.22%)	79.21%(±3.17%)	80.74%(±0.31%)	80.84%(±2.35%)	83.52%(±0.34%)	83.80%(±2.91%)	82.83%(±0.18%)	82.81%(±1.44%)
	LMS ∪ MCDO			97.25%(±0.09%)	**91.17**%(±1.75%)				
AUPRC	LMS	91.29%(±0.23%)	**73.47**%(±2.52%)	90.98%(±0.25%)	74.35%(±2.69%)	86.09%(±0.31%)	**64.54**%(±5.38%)	70.82%(±0.54%)	**50.25**%(±2.97%)
	LMS w/o features	87.73%(±0.20%)	72.00%(±2.67%)	87.34%(±0.27%)	73.70%(±2.91%)	81.11%(±0.51%)	64.29%(±5.38%)	65.49%(±0.42%)	48.26%(±3.31%)
	Entropy	49.16%(±0.59%)	48.48%(±5.75%)	48.16%(±0.61%)	48.24%(±5.99%)	47.25%(±0.48%)	47.91%(±4.53%)	35.16%(±0.33%)	35.29%(±2.88%)
	LMS ∪ MCDO			91.04%(±0.04%)	**74.39**%(±2.62%)				
Regression IoU_{adj}									
R^2	LMS	79.34%(±0.19%)	**66.69**%(±2.06%)	78.19%(±0.16%)	66.08%(±2.23%)	74.04%(±0.31%)	**61.57**%(±2.94%)	57.93%(±0.37%)	**48.84**%(±1.81%)
	LMS w/o features	75.78%(±0.18%)	64.91%(±1.87%)	74.91%(±0.18%)	65.31%(±2.21%)	70.78%(±0.30%)	61.51%(±2.96%)	54.73%(±0.19%)	47.62%(±1.64%)
	Entropy	51.45%(±0.37%)	50.88%(±2.74%)	48.54%(±0.57%)	48.21%(±4.25%)	50.90%(±0.45%)	50.30%(±3.27%)	39.51%(±0.28%)	39.33%(±2.57%)
	LMS ∪ MCDO			78.26%(±0.14%)	**66.13**%(±2.27%)				

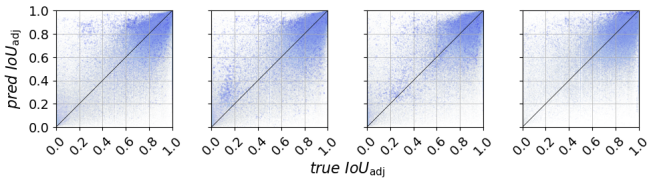


Fig. 3. True IoU_{adj} vs predicted IoU_{adj} for RangeNet++, SalsaNext, Cylinder3D on SemanticKITTI as well as Cylinder3D on nuScenes, from left to right.

an angular resolution of 0.33°. Every point cloud contains roughly 35K points. For our experiments we used the pretrained Cylinder3D with the recommended data split. We did not test RangeNet++ and SalsaNext since the corresponding pretrained models are not available.

The image projection is of size $(w, h) = (1090, 32)$. Computing the connected components for all samples of the 150 validation sequences yields approx. 1.5M segments. Excluding all small segments containing less than 10 points, i.e., $SP < 10$, reduces that number to 0.34M. Still, we retain 99% of the data in terms of points. We performed 10-fold cross validation where we always took 90% of the 150 sequences, i.e., 135 sequences, for training and the remaining 10%, i.e., 15 sequences for validation of the meta models. The results are presented in table I. 89.85% of all segments have an $IoU_{\text{adj}} > 0$. With the meta classification we achieve an accuracy of 91.00%, AUROC of 90.00% and AUPRC of 50.25%, see 'LMS' rows. For the meta regression we achieve $R^2 = 49.19\%$ for the validation data. The quality of predicting the IoU_{adj} is shown in fig. 3.

C. Metric Selection

So far, we have presented results based on all metrics from section II, indicated by LMS in table I. In order to analyze the impact of the metrics to the performance, we repeated the experiments for multiple sets of metrics.

a) Feature Measures: First, we tested the performance of the meta classification and regression model without the feature measures, i.e., the metrics based on the point cloud input features, see row 'LMS w/o features'. The performance in terms of ACC, AUROC, AUPRC and R^2 for all experiments are up to 2 percentage points (pp.) lower compared to when incorporating feature measures.

b) Entropy: Since the entropy is commonly used in uncertainty quantification, we tested all experiments with only using the mean entropy μE, see 'Entropy' rows. The performance for the meta classification is up to 12 pp. lower compared to LMS, for the meta regression R^2 decreases by up to 18 pp.

c) Bayesian Uncertainties: The projection-based SalsaNext model follows a Bayesian approach as already mentioned in section I: the LiDAR model provides a model (epistemic) and observation (aleatoric) uncertainty output for the point cloud's 2D image representation prediction, estimated by MC dropout (MCDO). To get these uncertainties we followed the procedure in [2]. This ends up in epistemic epi_l and aleatoric ale_l uncertainty values for each pixel position l. We compute the same aggregated measures as for the measures $M \in \mathcal{M}$. Adding these new metrics to the previous metrics LMS is refereed to as LMS ∪ MCDO. The additional Bayesian uncertainties do not improve the meta classification and regression performance significantly, see table I. We have not tested SalsaNext on nuScenes since the pretrained model is not available. For comparability of results, we only used publicly available pretrained models.

d) Greedy Heuristic: Inspired by fordward-stepwise selection for linear regression, we analyze different subsets of metrics by performing a greedy heuristic: we start with an empty set of metrics and iteratively add a single metric that maximally improves the performance – ACC for the false pos-

TABLE II

METRIC SELECTION USING A GREEDY METHOD THAT IN EACH STEP ADDS ONE METRIC THAT MAXIMIZES THE META CLASSIFICATION / REGRESSION PERFORMANCE IN TERMS OF ACC / R^2 IN %. ALL RESULTS, SEMANTICKITTI (TOP) AND NUSCENES (BOTTOM) ARE CALCULATED ON THE DATASET'S METRICS' VALIDATION SET.

	number of metrics	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	124
SemanticKITTI	RangeNet++																
	ACC	81.13	82.56	83.35	83.69	83.96	84.22	84.40	84.62	84.78	84.91	85.01	85.09	85.09	85.18	85.25	85.50
	Added	μV	P_{16}	$v\bar{F}^z_{in}$	P_{14}	P_{13}	$v\bar{F}^y$	P_{15}	μF^r_{bd}	μF^i	vF^z	N_1	P_{11}	vF^i	μD_{bd}	P_1	all
	R^2	56.74	58.20	59.01	59.87	60.72	61.57	62.07	62.69	63.24	63.70	64.06	64.43	64.72	64.91	65.15	66.69
	Added	μV	P_{14}	P_{16}	P_{17}	P_{15}	P_9	vF^y_{bd}	$v\bar{F}^z$	μF^i	P_{13}	P_3	vF^x	μF^r	N_{18}	SP	all
	SalsaNext																
	ACC	82.05	83.49	84.10	84.70	84.93	85.16	85.35	85.52	85.65	85.76	85.86	85.96	85.96	86.03	86.11	86.26
	Added	μV	P_{16}	P_{15}	P_{14}	vF^y_{bd}	P_{13}	$\mu\bar{D}$	vF^z	N_1	P_{12}	$\mu\bar{V}^z_{in}$	P_{10}	P_{11}	vF^i_{bd}	vF^x	all
	R^2	56.18	58.56	59.59	60.71	61.67	62.35	62.87	63.36	63.77	64.11	64.40	64.62	64.86	65.02	65.21	66.08
	Added	μD	P_{14}	P_{17}	P_{15}	P_{13}	vF^r	vF^z	P_{12}	P_1	P_9	P_2	$\mu\bar{V}^i$	vF^y	N_{10}	$\mu\bar{V}^z$	all
	Cylinder3D																
	ACC	86.13	86.76	87.19	87.41	87.60	87.75	87.84	87.93	88.02	88.08	88.17	88.26	88.26	88.29	88.33	88.37
	Added	μD	P_{15}	$v\bar{F}^r$	P_{17}	P_{11}	vD	P_{13}	P_{10}	$v\bar{F}^z$	N_1	SP	μF^r_{bd}	N_{15}	P_{14}	μF^i	all
	R^2	53.10	54.54	55.71	56.76	57.64	58.36	58.90	59.31	59.68	60.03	60.32	60.66	60.95	61.12	61.33	61.57
	Added	P_{14}	μE_{bd}	P_{17}	P_{15}	SP	$v\bar{F}^x$	vF^i	P_6	vF^y	N_9	P_{12}	vD	P_{13}	P_5	S	all

	number of metrics	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	118
nuScenes	Cylinder3D																
	ACC	90.06	90.27	90.44	90.53	90.61	90.67	90.73	90.8	90.85	90.87	90.89	90.91	90.91	90.93	90.95	91.00
	Added	μD	$\mu\bar{V}^r$	P_9	P_{16}	μF^x_{bd}	vF^r	P_{12}	P_4	$v\bar{F}$	$\mu\bar{V}^i$	N_{11}	P_{15}	μF^z_{bd}	P_3	vF^z_{bd}	all
	R^2	40.88	43.25	44.27	45.14	45.79	46.32	46.80	47.16	47.4	47.62	47.79	48.02	48.1	48.24	48.33	48.84
	Added	μV_{bd}	P_{16}	$v\bar{F}^r$	P_3	P_{13}	μF^r	P_4	μF^z	μF^y_{bd}	μF^i_{bd}	N_{14}	N_{10}	vF^z	P_{10}	P_1	all

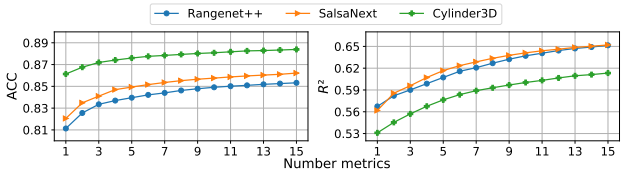


Fig. 4. Performance of the meta classification (left) and the meta regression (right) model on SemanticKITTI depending of the number of metrics, which are selected by the greedy approach.

itive detection and R^2 for the prediction quality estimation. We performed this greedy heuristic for both, meta classification and meta regression. The results in terms of ACC and R^2 are shown in fig. 4 (only for SemanticKITTI) and in table II. For the meta classification, we observe a comparatively big accuracy gain during adding the first 5 metrics, then the accuracy increases rather moderately. For the meta regression, this performance gain in terms of R^2 spreads wider across the first 10 iterations, before the improvement per iteration becomes moderate. Furthermore the results show that a small subset of metrics is sufficient for good models. We achieve nearly the same performance for both tasks with 15 metrics selected by the greedy heuristic compared to when using all metrics (LMS). Considering table II, the mean variation ratio μV and the mean probability difference μD in most cases constitute the initial choices. Furthermore, the mean probabilities $P_i,\ i \in \mathcal{C}$, are also frequently subject to early incorporation.

D. Confidence Calibration

The false positive detection is based on a meta classification model, which classifies whether the predicted IoU_{adj} is equal or greater than 0. In order to demonstrate the reliability of the classification model, we show that the confidences are well calibrated. Confidence scores are called *calibrated*, if the confidence is representative for the probability of correct classification, cf. [28].

The meta classification model estimates for each predicted segment the probability of being false positive, i.e., $IoU_{\text{adj}} = 0$. We group the probabilities for all meta classified segments of the validation data into 10 interval bins $(0.0, 0.1],\ (0.1, 0.2],\ \ldots,\ (0.9, 1.0]$. The accuracy of a bin is the relative amount of true predictions; the confidence of a bin is the mean of its probabilities. The closer the accuracy and the confidence are to each other, the more reliable is the corresponding classification model. This is visualized in a so-called reliability diagram. For the evaluation of calibration, we define the maximum calibration errors (MCE) as the maximum absolute difference between the accuracy and the confidence over all bins and the expected calibration errors (ECE) as a weighted average of the bins' difference between accuracy and confidence, where the weights are proportional to the number of elements per bin. Further details are given in [28].

The reliability diagrams and the MCE as well as the ECE for all previously discussed meta classification models are shown in fig. 5. The smaller the gaps, i.e., the closer the outputs are to the diagonal, the more reliable and well calibrated is the model. The MCE and ECE are between 5.26 – 10.68 and 0.62 – 1.63, respectively. The results indicate well calibrated and reliable meta classification models.

IV. CONCLUSION

In this work we presented our method LidarMetaSeg for segmentwise false positive detection and prediction quality estimation of LiDAR point cloud segmentation. We have shown that the more of our hand-crafted aggregated metrics we incorporate, the better the results get. This holds for all considered evaluation metrics – ACC, AUROC, AUPRC

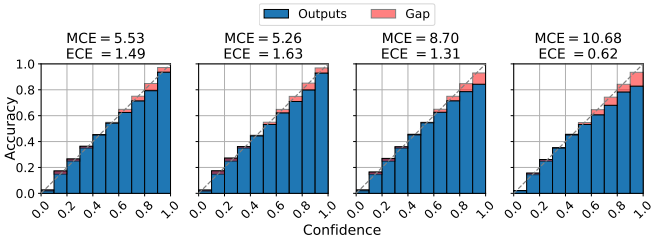


Fig. 5. Reliability diagrams with MCE and ECE for the meta classification model: RangeNet++, SalsaNext, Cylinder3D on SemanticKITTI as well as Cylinder3D on nuScenes, from left to right.

and R^2. Furthermore, the results show that adding Bayesian uncertainties (epistemic and aleatoric ones approximated by MC dropout) on top of our dispersion measures based on the softmax probabilities neither improves meta classification nor meta regression performance. We have demonstrated the effectiveness of the method on street scene scenarios and are positive that this method can be adapted to other LiDAR segmentation tasks and applications, e.g. indoor segmentation or panoptic segmentation.

REFERENCES

[1] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019.

[2] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving. *arXiv preprint arXiv:2003.03653*, 2020.

[3] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv preprint arXiv:2008.01550*, 2020.

[4] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. *arXiv preprint arXiv:2103.12978*, 2021.

[5] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. *IEEE Access*, 8:2847–2868, 2019.

[6] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[7] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[8] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[9] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *European Conference on Computer Vision*, pages 1–19. Springer, 2020.

[10] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[11] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[13] Onur Ozdemir, Benjamin Woodward, and Andrew A Berlin. Propagating uncertainty in multi-stage bayesian convolutional neural networks with application to pulmonary nodule detection. *arXiv preprint arXiv:1712.00497*, 2017.

[14] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[15] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3266–3273. IEEE, 2018.

[16] Matthias Rottmann, Pascal Colling, Thomas Paul Hack, Robin Chan, Fabian Hüger, Peter Schlicht, and Hanno Gottschalk. Prediction error meta classification in semantic segmentation: Detection via aggregated dispersion measures of softmax probabilities. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[17] Abhijit Guha Roy, Sailesh Conjeti, Nassir Navab, and Christian Wachinger. Inherent brain segmentation quality control from fully convnet monte carlo sampling. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 664–672. Springer, 2018.

[18] Robin Chan, Matthias Rottmann, Fabian Hüger, Peter Schlicht, and Hanno Gottschalk. Controlled false negative reduction of minority classes in semantic segmentation. 2020.

[19] Kira Maag, Matthias Rottmann, and Hanno Gottschalk. Time-dynamic estimates of the reliability of deep semantic segmentation networks. In *2020 IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2020.

[20] Matthias Rottmann and Marius Schubert. Uncertainty measures and prediction quality rating for the semantic segmentation of nested multi resolution street scene images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[21] Pascal Colling., Lutz Roese-Koerner., Hanno Gottschalk., and Matthias Rottmann. Metabox+: A new region based active learning method for semantic segmentation using priority maps. In *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,*, pages 51–62. INSTICC, SciTePress, 2021.

[22] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[23] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[24] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[25] Larissa T Triess, David Peter, Christoph B Rist, and J Marius Zöllner. Scan-based semantic segmentation of lidar point clouds: An experimental study. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1116–1121. IEEE, 2020.

[26] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[27] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[28] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.