LiDAR-MIMO: Efficient Uncertainty Estimation for LiDAR-based 3D Object Detection

by

Matthew A. Pitropov

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Author's Declaration

Statement of Contributions

This thesis involves joint work with several additional members within the lab. Chengjie Huang helped with the implementation of the softmax focal loss. Our co-op students Charles Zhang and Martin Ma contributed to the creation and testing of the CADC dataloader for OpenPCDet as well as the calibration error calculations. Spencer Delcore contributed to the uncertainty evaluation pipeline and validated many of the calculations that we use for the various metrics within the paper. Furthermore, Chengjie Huang, Dr. Vahdat Abdelzad, Professor Krzysztof Czarnecki and Professor Steven Waslander reviewed and edited the material for this thesis which is based off our publication that is being reviewed.

Abstract

The estimation of uncertainty in robotic vision, such as 3D object detection, is an essential component in developing safe autonomous systems aware of their own performance. However, the deployment of current uncertainty estimation methods in 3D object detection remains challenging due to timing and computational constraints. To tackle this issue, we propose LiDAR-MIMO, an adaptation of the multi-input multi-output (MIMO) uncertainty estimation method to the LiDAR-based 3D object detection task. Our method modifies the original MIMO by performing multi-input at the feature level to ensure the detection, uncertainty estimation, and runtime performance benefits are retained despite the limited capacity of the underlying detector and the large computational costs of point cloud processing. We compare LiDAR-MIMO with MC dropout and ensembles as baselines and show comparable uncertainty estimation results with only a small number of output heads. Further, LiDAR-MIMO can be configured to be twice as fast as MC dropout and ensembles, while achieving higher mAP than MC dropout and approaching that of ensembles.

Acknowledgements

I would like to thank my advisors Professor Krzysztof Czarnecki and Professor Steven Waslander for their indispensable guidance, support and feedback during my time exploring this research direction. I would also like to thank Dr. Vahdat Abdelzad, Chengjie Huang, Dr. Rick Salay, Dr. Sean Sedwards, Sun Sheng Gu, Harry Nguyen and Prarthana Bhattacharyya that have provided additional feedback throughout our many discussions.

Dedication

This thesis is dedicated to my family and friends for their support throughout my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Deep neural networks (DNNs) are used to create state of the art models for perception on autonomous vehicles (AV). One major issue at the moment is the lack of real-time and accurate information on perception uncertainty that would enable the AV to make correct decisions when faced with an uncertain scenario. For 3D object detection, the DNN produces the confidence score (max softmax) along with the position, dimensions and yaw of the prediction. Hendrycks and Gimpel [19] have shown that the mean softmax probability for incorrect predictions was shown to be very high on several datasets including 86% the MNIST dataset [27], thus these scores are not accurate estimates of the actual predictive uncertainty [19].

Bayesian Neural Networks (BNNs) [32] are a method for calculating reliable uncertainty estimates. Instead of a standard DNN where point estimates are used as network parameters, for BNNs a prior probability distributions is places on the network parameters. Bayesian inference is then used to calculate the posterior. For large network sizes used for AV perception tasks, Bayesian inference is computationally costly and intractable. As a result, most of the current research has concentrated on approximations that yield uncertainty estimates at a lower cost of computation. Deep ensembles [25] and Monte Carlo (MC) dropout [11] are well known approximation methods and have also been applied to object detection [9, 30]. These approximates follow the same concept which involves sampling the output from multiple models, each with their own approximation of the posterior distribution in order to calculate uncertainty as a statistic across outputs (e.g., variance of the outputs). Performing multiple network passes for each model is computationally expensive and therefore difficult to deploy on AVs where decisions must be made in real-time.

To overcome this issue, LiDAR-MIMO has been developed, an adaptation of the multi-input multi-output (MIMO) uncertainty estimation method [18] to the task of 3D object detection in LiDAR point clouds. MIMO is a method that has demonstrated state-of-the-art uncertainty estimates in image classification, but at a fraction of the computational cost of the existing methods. MIMO works by training a single network with multiple inputs and additional outputs (one per input). During testing the test image is fed in multiple times (one per input), the resulting outputs proved the distribution to estimate the predictive uncertainty.

Within the original MIMO method, the images are easily stacked as tensors increasing the number of channels of the input. Attempting to adapt this method to 3D LiDAR object detection is not as obvious. Combining multiple point clouds for input is computationally expensive and could negate any performance advantage of MIMO. We propose to instead combine the inputs at the feature level. This eliminates the bottleneck of combining point clouds or voxels and transfers it towards the middle of the network. Rather than combine raw inputs as in the original MIMO, we perform multi-input on the bird's-eye-view (BEV) feature maps which we found key for efficient uncertainty estimation for LiDAR-based 3D object detection. We credit this extension's effectiveness to the fact that the voxel feature network lacks the capacity to contribute significantly to uncertainty estimates.

The resulting LiDAR-MIMO design is trained with two different 3D object detector network architectures as well as two AV datasets and compared with MC dropout and ensemble models. Our results show that LiDAR-MIMO has comparable accuracy to the existing methods in terms of uncertainty estimation quality, as measured by scoring rules and calibration errors. However, t does so while being twice as fast as MC dropout and ensemble and having a greater detection performance (mAP) than both vanilla DNN (baseline) and MC dropout, as shown in Fig. 1.1.

1.2 Contributions

The main contributions of this thesis are as follows:

1. We are the first to adapt and show the effectiveness of the MIMO architecure in the task of LiDAR-based 3D object detection.

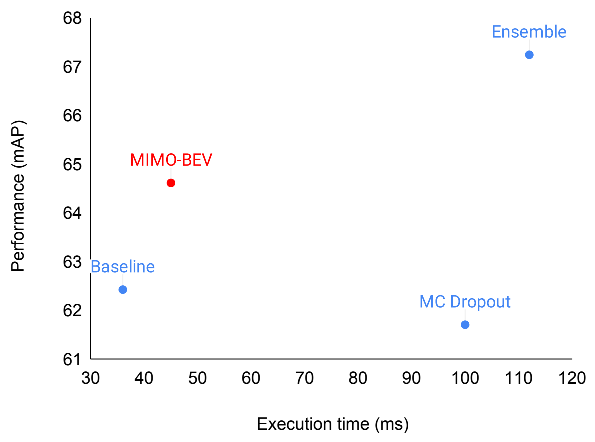

Figure 1.1: The detection performance (mAP) compared to the inference time (execution time) for the PointPillars models trained on the KITTI dataset

2. We modify the MIMO architecture to reduce the data processing time by combining inputs at the feature level instead of the raw inputs. This is key for real-time deployment on an AV. This approach also takes better advantage of the limited capacity of the network compared to the original MIMO.

3. We perform an in-depth evaluation of LiDAR-MIMO with multiple 3D object detectors and AV datasets showing results for scoring rules, calibration errors, mAP and execution time. LiDAR-MIMO is shown to have comparable uncertainty performance, higher mAP than MC dropout and only slightly slower than the original baseline detector. As a result, LiDAR-MIMO can be configured to be twice as fast as MC dropout or ensemble.

1.3 Overview

This thesis is structured as follows:

- Chapter 2 provide background knowledge on LiDAR-based 3D object detection and uncertainty for image classification as well as object detection.
- Chapter 3 introduces the LiDAR-MIMO object detector as well as the network modifications, clustering and merging methods used for all models.

- Chapter 4 discusses the experiments. This includes the datasets used, models trained, evaluation metrics and the results.
- Chapter 5 presents a conclusion along with potential future directions.

Chapter 2

Background

2.1 3D object detection for LiDAR point clouds

LiDAR-based 3D object detection has emerged as a pivotal component of autonomous driving systems, with point-based [36, 39], voxel-based [22, 44, 47, 26], and hybrid approaches showing steadily improvement [40]. Voxel-based approaches that exploit the bird's-eye view (BEV) projection, such as PointPillars (PP) [26] and SECOND (SC) [44], achieve competitive detection performance at low computational cost, which makes them attractive for deployment on AVs. Although hybrid approaches, like PV-RCNN [40], have higher mean average precision (mAP), they also add substantial computational cost. For example, PV-RCNN [40] achieves 82.97% mAP on the KITTI benchmark, compared to 80.29% mAP for SC, but SC runs 2x faster than PV-RCNN [37].

Fig. 2.1 highlights the main components of a voxel-based detector using PP and SC as examples. The input point cloud is first voxelized into pillar (PP) or cuboid (SC) voxels. These are fed into a voxel feature extraction (VFE) network, which is PointNet [35] for PP and sparse 3D convolutions for SC. The BEV projection of the resulting features, or BEV feature maps, are then passed to a backbone, which applies 2D convolutions. Finally, the backbone features are passed to a detection head, which outputs classes and bounding boxes of the detected objects. The BEV feature maps are sometimes referred to as a "2D pseudo image" [35].

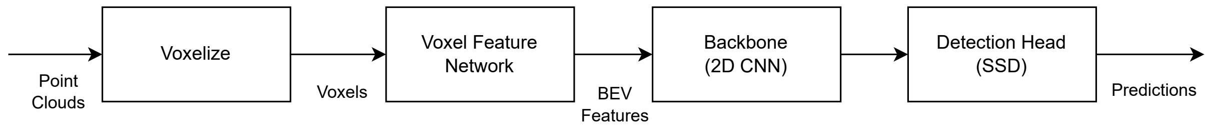

Figure 2.1: The network architecture of a voxel based 3D object detector.

2.2 Uncertainty estimation for classification

The predictive uncertainty of a classification network has two components [21]: epistemic uncertainty, which is due to the uncertainty over the model parameters caused by the limited availability of data; and aleatoric uncertainty, which is due to the inherent noise in the data itself. The predictive distribution output by a network should adequately reflect both sources of uncertainty, which is the case for Bayesian inference for NNs. However, since exact such inference is computationally intractable, practical realizations of BNNs that place a distribution on their weights are necessarily approximations, relying on making prior assumptions about the class of distributions for the weights and using different approximate inference methods, including variational Bayes [1] and Markov Chain Monte Carlo methods [32]. These methods are highly sensitive to hyper-parameters and difficult to scale to large datasets and network architectures [29].

A more practical class of approximate BNNs, which we discuss in the remainder of this section, rely on some form of model ensembling that samples models from the approximate posterior and combines their outputs to compute the resulting predictive distribution. For classification, the predictive distribution is simply the mean of the categorical distributions from each ensemble member [10]. For regression, each member is usually designed as a variance network [33], which is trained with two outputs, the mean and variance, using the so-called aleatoric regression loss [21]. The overall predictive distribution is then characterized by the mean and variance of the resulting mixture distribution [25]. We now briefly introduce ensembles and MC dropout, which are the two main practical methods for approximating BNNs found in literature, and MIMO, which aims at reducing the computational cost of these methods.

Deep ensembles [25]: A state of the art baseline in uncertainty estimation is to train multiple models with the same architecture and ensure their diversity by random initialization of weights and training data shuffling. In contrast to traditional ensembling methods like bootstrapping, each ensemble member is trained on the complete training dataset for maximum accuracy. Ensembling is shown to outperform MC dropout on uncertainty prediction metrics [25]. In particular, it shows much stronger diversity of outputs than MC dropout, since each model is trained independently [18]. Although easily parallelized, deep ensembles remain very costly in terms of both processing and memory use.

MC dropout [11]: MC dropout employs dropout both during training and inference and obtains multiple predictions by performing multiple forward passes with different dropout network realizations for the same input. Normally dropout layers are added throughout the network as a regularization method to reduce over-fitting [42]. Interestingly, MC dropout has been shown to be equivalent to approximate Bayesian inference in deep Gaussian processes [11]. MC dropout can be thought as a form of implicit ensembling with a single network, but it still requires multiple passes, which can be computationally prohibitive for robotic vision applications.

MIMO [18]: Similar to MC dropout, a multi-input multi-output (MIMO) network represents an ensemble as a single network, but unlike MC dropout, it requires only a single forward pass during inference. The main idea is to stack multiple inputs as a combined input into a single network with multiple copies of its classification head, one per input. The network is trained by stacking different inputs and then giving each classification head the ground truth label of the corresponding input for its loss function. During inference, instead of stacking different inputs, the same input is stacked to satisfy the input tensor size requirement. Each head makes its own prediction for the same input, which results in the desired sample set of outputs. MIMO relies on the fact that modern DNNs are overparameterized and uses the available network capacity to fit multiple independent subnetworks. Because each output head is trained by independently sampled inputs, given sufficient capacity, the MIMO predictions can match the diversity of a deep ensemble [18].

When compared with MC dropout and ensembles, the MIMO method was demonstrated to provide a prediction time similar to a vanilla deterministic network. MIMO achieved this while outperforming the classification accuracy of MC dropout, and approaching that of an ensemble [18]. It also had Negative Log-Likelihood (NLL) and Expected Calibration Error (ECE) [31] scores comparable to deep ensembles, indicating uncertainty estimation of similarly high quality. Recently, MIMO has been adapted for the task of 2D object detection showing similar performance results [5].

2.3 Uncertainty estimation for object detection

Uncertainty estimation has recently also attracted interest in the context of object detection. For this task, uncertainty estimation methods are used both for object classification and bounding box-regression, which results in bounding boxes being represented by probability distributions. In contrast to classification, object detection also needs to deal with multiple object hypotheses, and thus steps such as non-maximum suppression (NMS) need to be refined or extended to deal with bounding boxes as probability distributions.

Most closely related to our work, Feng et al. [7] propose a probabilistic LiDAR-based 3D vehicle detector with dropout layers and a variance regression output for predicted variances. They find that adding the variance regression output improves vehicle detection accuracy. The authors did not evaluate uncertainty estimation, but the results show that the occlusion level and vehicle distance correlate with the aleatoric uncertainty, and the vehicle detection accuracy correlates with the epistemic uncertainty.

Similarly, Zhong et al. [46] modify the SECOND architecture with a variance regression output and capture aleatoric uncertainty via an adapted aleatoric loss function. The loss improves the uncertainty prediction for angular parameters by using the von Mises distribution.

Harakeh and Waslander [17] conduct an in-depth survey and evaluation of predictive uncertainty for deep 2D object detection architectures using different aleatoric regression losses, MC dropout, and ensemble methods. To evaluate the methods, they use mAP, calibration error, and scoring rules. The results highlight i) the importance of scoring rules for capturing the reliability of output distributions, ii) issues with NMS while seeking output sample sets, and iii) computational and performance limitations of ensemble and MC dropout methods. Our work follows in this vein of rigorous evaluation of predictive uncertainty, and addresses the high computational cost of MC dropout and ensembles.

We are unaware of any work adapting MIMO for uncertainty estimation for 3D object detection or LiDAR inputs.

Chapter 3

LiDAR-MIMO 3D object detection

Fig. 3.1 depicts our proposed adaptation of the MIMO method to LiDAR-based 3D object detection, with the components of the underlying 3D object detector (e.g, PointPillars or SECOND) in black and the extensions required by LiDAR-MIMO in red. We propose a design variant for combining multiple inputs named MIMO-BEV. In MIMO-BEV, we stack BEV feature maps of point cloud inputs as multiple inputs. In the following sections, we describe the input combination, the loss functions, and the clustering of the predictions from the multiple heads.

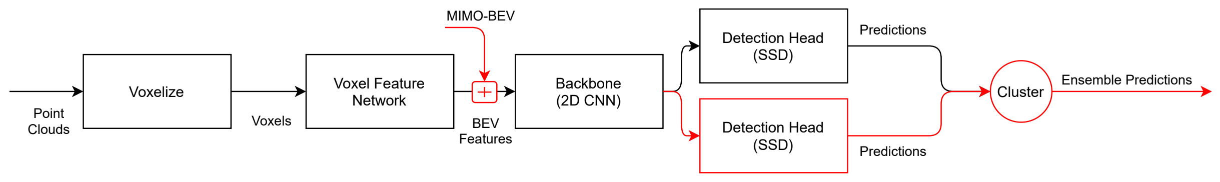

Figure 3.1: The network architecture of a LiDAR-MIMO 3D object detector with additions highlighted in red

3.1 Input combination

For the MIMO-BEV variant, the main idea is that the equivalent of stacking images for MIMO in image classification is stacking the BEV pseudo image in 3D object detection. Fig. 3.2 depicts the MIMO-BEV architecture for training. In step 1, one point cloud for each detection head is individually voxelized, followed by step 2 where those voxels are sent into the VFE network. Note that the voxelization and VFE functions take one input point cloud and are called separately for each input. The input point clouds can be from different frames or the same frame, with the likelihood depending on the chosen MIMO parameters. These parameters are the number of heads, input repetition, and batch repetition and are discussed in Sec. 4. At this point each point cloud has been converted into separate BEV pseudo images (feature maps), which are then stacked together in step 3 as input for the Backbone. During testing, we instead follow the steps in Fig. 3.3. A single point cloud is voxelized in step 1 and sent through the VFE network for step 2. The output, which is the pseudo image, is duplicated multiple times depending on the number of heads in step 3. Then, these the pseudo images are stacked in step 4 and used as input to the backbone.

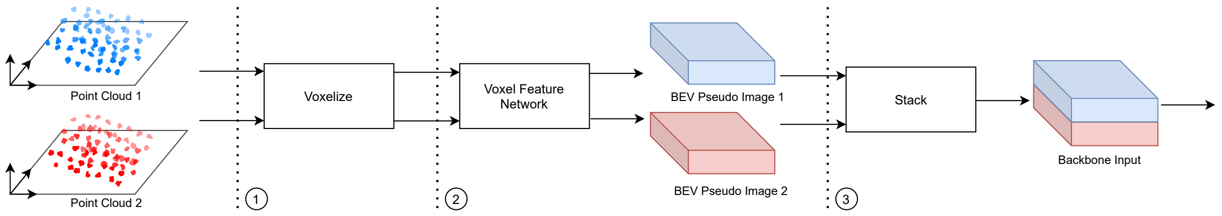

Figure 3.2: Pseudo image stacking during training for MIMO-BEV with two inputs

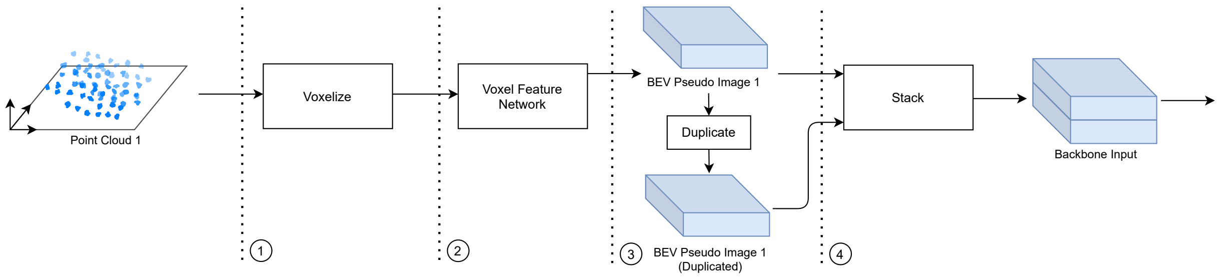

Figure 3.3: Pseudo image stacking during testing for MIMO-BEV with two inputs

We also experimented with an additional variant that combines point clouds before the VFE. This variant, called MIMO-ID, adds an additional head ID feature to each point to allow the network to differentiate among the inputs. In contrast to MIMO-BEV, MIMO-ID can also fit subnetworks into the VFN, rather than just the Backbone. For the MIMO-ID variant, we combine the point clouds before the VFE network and add an additional feature to each point in the point cloud to separate the point clouds from one another. Fig. 3.4 depicts the MIMO-ID architecture for training. In step 1, one point cloud for each detection head is individually voxelized. In step 2, a head ID feature is added to each point in the voxels. The final step is to merge the voxels as the input to the VFE network. This combination method of adding head IDs allows the detection heads to focus on features from their corresponding point cloud. It also ensures that the voxels are filled evenly by all inputs. During testing, we found that the voxel merging strategy was slow and took a different approach to slightly increase the data processing speed. We follow the steps in fig. 3.5. A single point cloud is duplicated for each detection head in step 1, followed by the additional head ID features being added to the points. In step 2 we merge the point clouds into a single cloud by simply concatenating their arrays. In step 3 the point cloud must be shuffled. This is to resolve the problem of voxels being filled from only the first point cloud in the point array. In the final step, the point cloud is voxelized and send to the VFE network. During training and testing, the maximum number of points per voxel must be multiplied by the number of detection heads, compared to the original implementation.

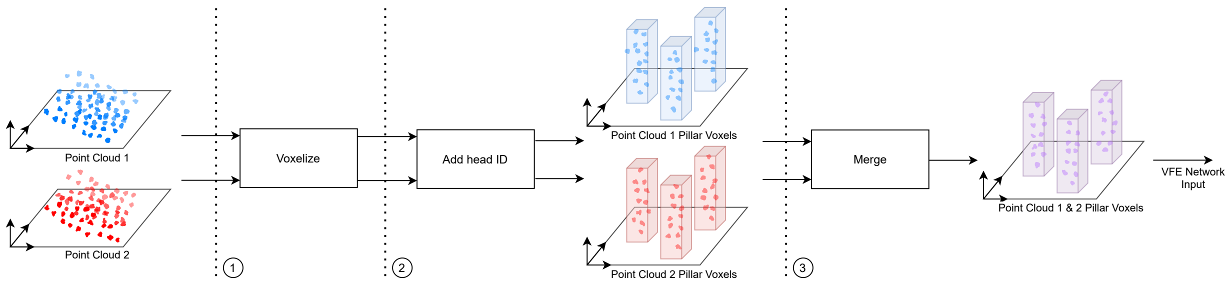
Point Cloud 1
Point Cloud 2
Voxelize
Add head ID
Point Cloud 1 Pillar Voxels
Point Cloud 2 Pillar Voxels
Merge
Point Cloud 1 & 2 Pillar Voxels
VFE Network Input
1
2
3

Figure 3.4: Voxel merging during training for MIMO-ID with two inputs

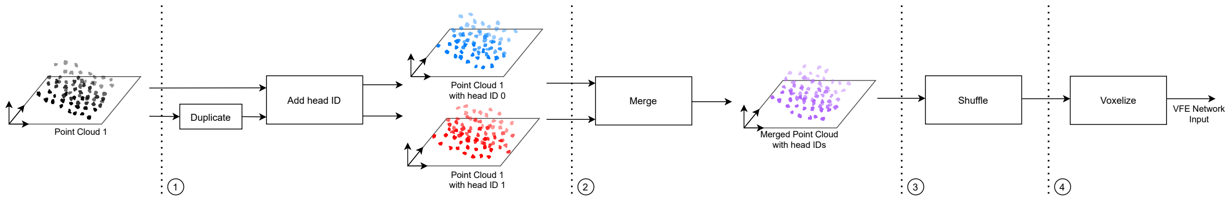
Point Cloud 1
Duplicate
Add head ID
Point Cloud 1 with head ID 0
Point Cloud 1 with head ID 1
Merge
Merged Point Cloud with head IDs
Shuffle
Voxelize
VFE Network Input
1
2
3
4

Figure 3.5: Point cloud merging during training for MIMO-ID with two inputs

3.2 Loss functions

Classification loss: We use a softmax version of the original focal loss [28], which allows us to use uncertainty evaluation metrics such as the Brier Score that require a probability distribution over all classes. This is different from the default choice in PointPillars and SECOND implementations, which use one sigmoid per class.

Regression on linear parameters: To obtain variances for the bounding box regression values, we add an extra convolution layer as output to the detection head(s) for heteroscedastic regression [21]. For such a regression, the noise is data-dependent and must be learned as part of the loss function, see Equation (3.1) [7, 21]. The ground-truth bounding box $\mathbf{y}_{gt}$ and predicted bounding box $\mathbf{u}_{\mathbf{x}}$ parameters include seven regression variables: centroid position, x, y, z, length, width, height and orientation. The log variance, $\lambda = \log \sigma^2$, for each parameter is regressed for numerical stability, to avoid the possibility of dividing by zero [21]. When data has high uncertainty, the first term becomes 0 and the model's loss only contains the second term. This causes the loss to be less affected by data with higher uncertainty.

$$L_{var} = \frac{1}{2} \exp(-\lambda) (\mathbf{y}_{gt} - \mathbf{u}_{\mathbf{x}})^2 + \frac{1}{2}\lambda \tag{3.1}$$

Regression on angular parameters: For regression on linear parameters, the loss function is derived from the Negative Log-Likelihood (NLL) of the Gaussian density function. In contrast, for regression on angular parameters, which are periodic, a periodic approximation of the Gaussian distribution should be used [46]. Thus, we use Equation (3.2), derived from the NLL of the von Mises PDF (see [46] for details).

$$\begin{gathered} L_{var_\theta} = \log I_0(\exp(-\lambda)) - \exp(-\lambda) \cos(\theta - \theta_t) \\ + ELU(\lambda - \lambda_0) \end{gathered} \tag{3.2}$$

3.3 Output clustering and merging

In the MIMO architecture, it is necessary to merge outputs to obtain the final predictions (see Fig. 3.1). This is straightforward for image classification, because of the single object per input to be classified. In contrast, for object detection, we may have multiple objects per input and thus need to cluster and merge the detections of the same object from multiple heads. To group predictions we cluster the objects that have a minimum of 0.5 IoU. We then use the consensus clustering method, shown to perform best compared to the affirmative (at least one) and unanimous (all members must agree) methods [4]. Consensus clustering requires more than the number of outputs divided by 2 in order for the cluster to be valid. To merge predictions, we used the mean method. Roza et al. [38] implemented several different box merging strategies, and showed significant variations on sample-based uncertainty estimates resulting from different strategies. Based on their results, we used the mean method, which had the best performance overall. Orientation angles need a special consideration, however, since they could differ by 180° in the outputs. To handle this case, we instead take the angle from the box with the highest confidence to remove the negative effect of mean methods on bimodal distributions. For example, two clustered predictions with 0° and 180° would result in worse prediction with a mean angle of 90°.

Chapter 4

Experiments

In this section, we evaluate our proposed LiDAR-MIMO architecture against other uncertainty estimation methods on publicly available AV datasets. We demonstrate that LiDAR-MIMO produces on par uncertainty estimates while having lower inference latency, which is paramount for safety-critical and resource-constrained applications such as AVs.

4.1 Datasets

We experiment on two different datasets: KITTI [12] and CADC [34]. The KITTI dataset is a widely-used multimodal dataset for autonomous driving in clear weather. In contrast, the recently released CADC dataset focuses on snowy weather conditions. It makes detection challenging due to snowfall, leading to higher uncertainty results.

The full training set of each dataset is used to train each model. For mAP and partition counts, we report the results the on the full validation (val) set. For uncertainty evaluation, we follow Feng et al. [8] and split the val set equally into the recalibration (recal) set and the evaluation (eval) set. The exact data split was not given by Feng et al. [8] so we split the data by using the first and second half of the randomly shuffled frames. Due to the large number of samples in the dataset we do not perform cross-validation and use the same split of data for our experiments [15, Chapter 5.3]. The recal set is used to determine score thresholds and calibrate the models. The eval set is then used to test the calibrated model by calculating the scoring rules and calibration errors.

4.2 Models

We evaluate LiDAR-MIMO against MC dropout and ensembles on two detectors, PointPillars (PP) and SECOND (SC). We use the PP and SC implementations provided in OpenPCDet [43], and extend them with MC dropout, ensembling, and MIMO-BEV. We refer to the resulting detectors for MC dropout, ensemble and MIMO-BEV as *multi-output*.

Baseline: The baseline models for evaluating detection performance use the original PP and SC implementations. For KITTI, we use the PP and SC models provided with OpenPCDet. For CADC, we extend OpenPCDet with a data loader for CADC and train our own baseline models, with a batch size of 6. All baseline models are trained with the sigmoid focal loss for 80 epochs. We found CADC to be challenging for our MC dropout and MIMO-BEV models resulting in 2-3% lower mAP compared to the baseline. We believe that this is due to the voxel parameters set for 360° detection for the CADC dataset. Consequently, we limit our CADC results to the SC models and include only include mAP results for PP models.

Multi-output: Each multi-input model shares the same voxel-feature network, backbone, and head from the original PP and SC, with some exceptions for MC dropout and MIMO-BEV as described shortly. Further, all multi-output detectors use the softmax focal classification loss, smooth L1 regression losses, and the previously described aleatoric regression losses, and share the same clustering and merging stage. Each multi-input model is trained for an extra 40 epochs compared to the baseline models in order to have the models converge with the new loss functions.

MC dropout: We create a variant of the backbone by inserting a dropout layer, with 0.5 probability, after the ReLU of each deconvolution block. During inference, we keep the dropout layers active for MC dropout. This model is trained with a batch size of 6 for 120 epochs.

Ensemble: We train four unique models to form our ensemble. Each model is trained with a different random seed for frame order shuffling, for a batch size of 6 with 120 epochs.

MIMO-BEV: MIMO-BEV requires additional code for pseudo images combination and adding multiple heads. There are three hyper parameters for MIMO: number of heads, input repetition (IR) and batch repetition (BR). We selected the number of heads to be 2 as the base networks have low capacity. IR is used to select a percentage of frame groupings in a batch to have matching frame, while BR duplicates each frame in a batch for smoother training. Since we are using a low batch size we were able to train with an IR of 0% and a BR of 0. This model is trained with a batch size of 3 with 120 epochs. We include results for various batch sizes and input repetitions.

Tab. 4.1 specifies the runtime parameters for the evaluated models. The number of forward passes is the number of times that a model is run for input to output(s). The number of outputs is the number of NMS outputs after all forward passes are run. For example, an MC dropout model is run 4 times to get 4 outputs, and a MIMO model is run once for 2 outputs, since it is set for 2 heads. We use two heads since our experiments have shown that PointPillars and SECOND do not have sufficient capacity to accommodate three or more independent subnetworks, and adding more capacity would come with increased runtime. This is consistent with the original MIMO work [18], where the optimal number of subnetworks for the studied image classification networks is 2-3.

Table 4.1: Number of forward passes, outputs and minimum cluster size used for models in our evaluation

Model	Forward Passes	Outputs	Minimum Cluster Size
Baseline	1	1	-
MC dropout	4	4	3
Ensemble	4	4	3
MIMO-BEV	1	2	2

4.3 Evaluation metrics

We evaluate models based on the detection performance, inference time, and uncertainty estimates for both classification and regression.

Detection: 3D object detection algorithms are generally evaluated using the average precision (AP) metric, first introduced in the PASCAL VOC 2007 object detection challenge [6]. The calculation of AP is based on the area under the precision-recall curve calculated for 40 recall points. Fig. 4.1 contains an example of a PR curve calculated with the output from a baseline PointPillars model trained on the KITTI dataset. Precision is the percentage of correct predictions and is calculated by the number of correct predictions divided by the total number of predictions. Recall is the percentage of ground-truth objects correctly predicted, it is calculated by the number of correct predictions divided by the number of ground-truth objects.

Scoring rules: For evaluating uncertainty, we use scoring rules [13]. A scoring rule $S(p, q)$ compares the probability distribution p with the true distribution q. A scoring rule can be proper or strictly proper as well as local or non-local. A proper scoring rule $S(p, q)$

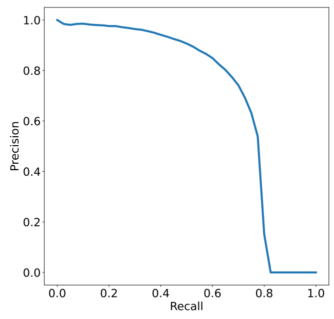

Figure 4.1: An exemplary PR curve of the baseline PointPillars model trained on the KITTI dataset.

is maximised when $p = q$ or other values such that $S(q, q) \geq S(p, q)$. A strictly proper scoring rule $S(p, q)$ is maximised only when $p = q$ such that $S(q, q) = S(p, q)$. A local scoring rule evaluates the distribution only at the actual output (e.g. max softmax), while a non-local rule evaluates other features.

Calibration error: A well-calibrated model for classification with 100 predictions that have 0.8 confidence score should be correct 80% of the time. Any differences can be used to calculate the calibration error (CE) in order to determine the uncertainty calibration quality. To create a calibration plot [16] as displayed in Fig. 4.2, first probabilistic predictions are run for all data samples followed by dividing these predictions into T intervals within the horizontal axis and lastly calculating the empirical probability by comparing these predictions to the ground-truth. The red bars display the error and the red colour's alpha value is set based on the number of predictions in each bin. Rather than take the sum of all bins, CE variations include the maximum CE which is only the value of the bin with the highest error as well as the Expected CE (ECE) which weights each bin by the number of objects within.

Classification uncertainty: We evaluate this uncertainty based on the softmax distribution extracted for each prediction using two metrics: negative log-likelihood (NLL) and Brier score, which are both proper [13].

The NLL score is a local and proper scoring rule which only evaluates the softmax distribution at the ground truth label. A lower NLL score indicates a better fitting predictive distribution for that specific ground truth label. In equation (4.1), there are N predictions and C classes with $\hat{s}_{n,c}$ being the predicted classification score for the c^{th} class and $y_{n,c}$ as the corresponding one hot encoded ground-truth label. The range of the NLL score is from

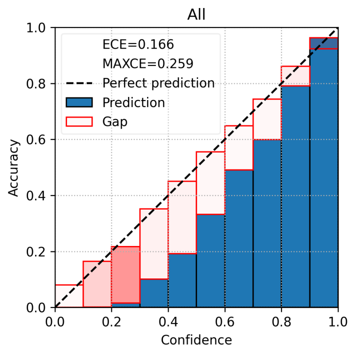

Figure 4.2: An exemplary reliability diagram for a PointPillars ensemble model trained on the KITTI dataset.

$-\infty$ to $+\infty$.

$$\text{NLL} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}\log(\hat{s}_{n,c}) \tag{4.1}$$

The Brier score [2] is a non-local and strictly proper scoring rule which evaluates the entire softmax distribution by taking the squared error between a predictive probability from the network and its one-hot encoded ground truth label. It is used to measure the accuracy of probabilistic predictions specifically in classification. In equation (4.2), the Brier Score is the squared error between a predictive probability from the network and its one-hot encoded ground truth label. There are N predictions and C classes with $\hat{s}_{n,c}$ being the predicted classification score for the c^{th} class and $y_{n,c}$ as the corresponding ground truth label. The range of the Brier Score is from 0 to 2 for multiple classes with a lower score representing more accurate uncertainty estimation.

$$\text{BS} = \frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\left(\hat{s}_{n,c} - y_{n,c}\right)^2 \tag{4.2}$$

To evaluate the classification calibration, we use an extension of Average Calibration Error (ACE) called Marginal Calibration Error (MCE) [24]. ACE calculates the absolute error averaged over all score intervals equally and only considers the uncertainty calibration quality on the object's ground-truth class. In contrast, MCE measures the uncertainty of a classifier's predicted distribution over all classes of interest.

Regression uncertainty: For regression parameters, we evaluate uncertainty based on the predicted means and variances for each prediction that can be matched to a ground-truth object. We use NLL and energy scores [14] as scoring rules.

The NLL is a proper scoring rule that is also local. In (4.3), $\boldsymbol{z}_n$ is defined as the ground-truth regression target, $\boldsymbol{\Sigma}(\boldsymbol{x}_n, \boldsymbol{\theta})$ and $\boldsymbol{\mu}(\boldsymbol{x}_n, \boldsymbol{\theta})$ are the mean and variance of the output prediction. The first term will increase as the mean predicted regression values diverge from the ground-truth target. The second term is a regularization term that is included to prevent the network from predicting infinite uncertainty causing the loss to become 0 [21].

$$\mathrm{NLL} = \frac{1}{2N} \sum_{n=1}^{N} \hat{\boldsymbol{z}}_n^\top \boldsymbol{\Sigma}(\boldsymbol{x}_n, \boldsymbol{\theta})^{-1} \hat{\boldsymbol{z}}_n + \log \det \boldsymbol{\Sigma}(\boldsymbol{x}_n, \boldsymbol{\theta}) \tag{4.3}$$
$$\text{where } \hat{\boldsymbol{z}}_n = \boldsymbol{z}_n - \boldsymbol{\mu}(\boldsymbol{x}_n, \boldsymbol{\theta})$$

The energy score is a strictly proper and non-local score, analogous to the Brier score for classification. It is derived from energy distance. To calculate it, we use an efficient Monte-Carlo approximation for multivariate Gaussians in (4.4) [14]. There are N objects in the test set with M ensemble members. The $\mathbf{z}_n$ is ground truth while $\boldsymbol{z}_{n,i}$ is the i^{th} i.i.d sample from $N(\mu(x_n, \theta), \Sigma(x_n, \theta))$. It minimizes similarly to NLL, however, unlike NLL it will also penalize high entropy distributions. This results in better calibrated and lower entropy predictive distributions compared to NLL [17].

$$\mathrm{ES} = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{M} \sum_{i=1}^{M} \|\mathbf{z}_{n,i} - \boldsymbol{z}_n\| - \frac{1}{2(M-1)} \sum_{i=1}^{M-1} \|\mathbf{z}_{n,i} - \mathbf{z}_{n,i+1}\| \right) \tag{4.4}$$

For the calibration error, we use a method created by Kuleshov et al. [23], which is an extension to calibration methods for classification. Instead of the confidence scores being used to place predictions into bins based on thresholds, the CDF is used to place predictions. The calibration error is then calculated as the difference between the empirical frequency of how many predictions are in each bin and the actual number of objects in each confidence interval.

Disagreement: Two distance metrics are used by Havasi et al. [18] in order to measure the diversity of the multi-output models. The first is disagreement, which is calculated by counting the percentage of how many models disagree on the top class. The second is the Kullback–Leibler divergence which compares each class of the model outputs. We adapted these versions for object detection as well as introduced new distance calculations for regression values. For disagreement we calculate the percentage of predictions where the foreground class is different (ignoring background class), if the cluster is missing predictions then that is counted as a disagreement. For the Kullback–Leibler divergence, if a cluster is missing a prediction we create a softmax distribution to use with the foreground classes to 0.1 and the background class set to 0.7. This softmax distribution of $[0.1, 0.1, 0.1, 0.7]$ is a non-confident prediction at the threshold for removal. Lastly for regression we compare the position difference, difference in diagonal length of the dimensions as well as angle difference.

Detection partitions: We focus on three partitions in order to see clear differences in the uncertainty metrics [20, 17]. True Positives (TP) are predictions that match a ground-truth (GT) box with an IoU over the TP IoU threshold. Mislocalized false positives (FP_{ML}) are predictions that match with a GT box, but their IoU is below the TP IoU threshold and higher than 0.1. The last partition is background false positives (FP_{BG}), where the prediction has a lower than 0.1 IoU with all GT boxes and thus it is highly likely that the prediction should be background.

4.4 Results

4.4.1 The inference time

Tab. 4.2 shows the inference time of each uncertainty estimation method for PointPillars and the KITTI dataset. All timings were calculated using a computer with an i7-8700K CPU and Titan Xp GPU. Data processing time is the time to load the point cloud and perform voxelization. For each model except MC dropout, the calculation for total time is data processing time added to the number of forward passes multiplied by the time for data to pass through the network. For the MC dropout model an efficient implementation can have the VFE network output cached and reused for each forward pass that encounters the dropout layers. Due to this caching, the calculation adds the data processing time to the VFE time, which is then added to the number of forward passes multiplied by the backbone and heads time. As seen in Tab. 4.2, MIMO-BEV is 60% faster than ensemble and 55% faster than the MC dropout. Despite our attempts to decrease the data processing time during testing, MIMO-ID requires an additional 30 ms. The larger voxels also cause an increase in time for the VFE network, although it is only a 1 ms increase.

Table 4.2: The inference time (in *ms*) for the PointPillars model trained on the KITTI dataset

Model	# Fwd Passes	Data Processing	VFE	Backbone + Heads	Total Time ↓
		1 Fwd Pass			
Baseline	1	16	4	16	36
MC dropout	4	16	4	20	100
Ensemble	4	16	4	20	112
MIMO-BEV	1	16	4	25	**45**
MIMO-ID	1	46	5	25	76

4.4.2 Average precision and detection partitions

For each network architecture and dataset, we show separate tables with mAP and counts for individual detection partitions.

Tab. 4.3 and Tab. 4.4 contain the results for their respective models trained on the KITTI dataset. In both cases, the MIMO-BEV model exceeds the mAP score of the MC dropout model, and the baseline model, while approaching the Ensemble model. The MIMO models have higher FP_{BG} than the baseline model; however, their FP_{ML} is lower than all others except ensemble. The ensemble models have the largest increase in mAP after clustering; for example, the individual member outputs scored 62-63% mAP for PointPillars on KITTI. The ensemble also has the lowest FP_{BG} count. Fig. 1.1 plots mAP and execution time for the PointPillars models trained on KITTI. The smaller improvement of MIMO over the SECOND baseline may indicate that the latter makes better use of its available capacity than PointPillars.

Table 4.3: The mAP and detection partitions for the PointPillars models trained over the KITTI dataset

Model	mAP (%) ↑	TP	FP_{ML}	FP_{BG}
		Counts Per Partition		
Baseline	62.42	8128	4204	47028
MC dropout	61.63	7998	4348	44631
Ensemble	**66.97**	**8421**	**3465**	**32006**
MIMO-BEV	64.24	8287	3909	56858

Tab. 4.5 shows the results for the SECOND models trained on the CADC dataset. The MIMO-BEV model outperforms the MC dropout model in mAP score, but under-performs

Table 4.4: The mAP and detect partitions for the SECOND models trained over the KITTI dataset

Model	mAP (%) ↑	TP	FP_{ML}	FP_{BG}
		Counts Per Partition		
Baseline	65.37	8397	3837	36713
MC dropout	65.64	8385	3813	33786
Ensemble	**68.94**	**8642**	**3198**	**24295**
MIMO-BEV	66.00	8529	3555	42148

compared to the baseline model. Despite a lower mAP, the MIMO models has more TP and fewer FP predictions. The different effect on FP counts between KITTI and CADC datasets for SECOND could be due to differences in the parameters for voxel sizes and maximum points per voxel in order to accommodate 360° predictions in CADC.

Table 4.5: The mAP and detection partitions for the SECOND models trained over the CADC dataset

Model	mAP (%) ↑	TP	FP_{ML}	FP_{BG}
		Counts Per Partition		
Baseline	55.72	17645	7809	29204
MC dropout	52.06	17154	8132	24904
Ensemble	**58.38**	**18486**	**6435**	**17182**
MIMO-BEV	53.04	18051	7326	25977

In Tab. 4.6 we show the mAP and detection partition counts for the MIMO-ID models trained on the KITTI dataset. The MIMO-ID PointPillars model trained on KITTI performs only 0.32% worse than the MIMO-BEV model. The MIMO-ID SECOND model performs better than the MIMO-BEV version by 0.72%. In both MIMO-ID models, the FP_{BG} is lower showing that point cloud combination may be better at removing FP_{BG} detections. The number of FP_{BG} detections is equivalent to or lower than the baseline models.

4.4.3 Uncertainty evaluation metrics

The uncertainty results are laid out in Tab. 4.7, Tab. 4.8, and Tab. 4.9. Scores are averaged over IoU thresholds from 0.5 to 0.95, with a 0.05 increment. For each threshold we follow Harakeh and Waslander [17] by calculating a score threshold that maximizes the F1 score

Table 4.6: The mAP and detection partitions for the MIMO models trained over the KITTI dataset

Model	mAP (%) ↑	TP	FP_{ML}	FP_{BG}
		Counts Per Partition		
MIMO-BEV (PP)	64.61	8332	3857	56988
MIMO-ID (PP)	64.29	8250	3905	53593
MIMO-BEV (SC)	65.45	8534	3581	44113
MIMO-ID (SC)	**66.17**	**8586**	**3497**	**36509**

in order to remove low scoring predictions. Temperature scaling [16] is also performed per class at each score threshold to calibrate the predictions. The results are averaged over three models.

Table 4.7: Uncertainty evaluation for the PointPillars models trained over the KITTI dataset

Model	NLL (Cls) ↓			Brier Score ↓			NLL (Reg) ↓		Energy Score ↓		MCE (Cls) ↓	CE (Reg) ↓
	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	TP	FP_{ML}		
MC dropout	**0.347**	**0.414**	1.050	**0.190**	**0.238**	0.803	-4.858	-1.953	0.449	0.686	0.223	0.065
Ensemble	0.351	0.428	1.003	0.194	0.246	0.765	**-4.957**	**-2.256**	**0.434**	**0.664**	**0.200**	0.067
MIMO-BEV	0.363	0.439	**0.980**	0.199	0.256	**0.758**	-4.940	-2.178	0.451	0.684	0.202	**0.064**

Table 4.8: Uncertainty evaluation for the SECOND models trained over the KITTI dataset

Model	NLL (Cls) ↓			Brier Score ↓			NLL (Reg) ↓		Energy Score ↓		MCE (Cls) ↓	CE (Reg) ↓
	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	TP	FP_{ML}		
MC dropout	**0.326**	**0.383**	1.116	**0.171**	**0.214**	0.861	**-5.681**	-1.927	**0.284**	0.570	0.227	**0.058**
Ensemble	0.349	0.399	1.075	0.190	0.225	0.821	-5.632	**-2.286**	0.301	0.573	0.209	0.064
MIMO-BEV	0.346	0.400	**1.051**	0.187	0.227	**0.812**	-5.472	-2.224	0.328	**0.568**	**0.205**	0.067

Table 4.9: Uncertainty evaluation for the SECOND models trained over the CADC dataset

Model	NLL (Cls) ↓			Brier Score ↓			NLL (Reg) ↓		Energy Score ↓		MCE (Cls) ↓	CE (Reg) ↓
	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	TP	FP_{ML}		
MC dropout	**0.413**	0.586	0.743	**0.253**	0.378	0.554	-3.885	1.534	0.420	0.930	**0.190**	**0.070**
Ensemble	0.475	0.623	**0.688**	0.300	0.410	**0.504**	-4.090	**-0.426**	0.429	**0.797**	0.196	0.077
MIMO-BEV	0.414	**0.564**	0.736	0.254	**0.364**	0.545	**-4.170**	-0.110	**0.379**	0.838	**0.190**	0.072

MC dropout often outperforms the other models for classification NLL and Brier Score in the TP and FP_{ML}, while MIMO-BEV performs well for FP_{BG}s. The MC dropout has predictions with higher confidence leading to lower scores. The MIMO-BEV model outputs lower confidence predictions, causing a better score for FP_{BG}s.

For the regression scores, the ensemble model performs well on the KITTI dataset, with some best scores being taken by MC dropout and MIMO-BEV. Once again the scores are low and are similar. For CADC the MIMO-BEV model performs best for TPs, while the lowest $\mathrm{FP}_{\mathrm{ML}}$s scores are for the ensemble model.

The calibration errors are similar among all the models. The MCE values tend to be lowest for MIMO-BEV and ensemble, while the CE is lowest for MC dropout and MIMO-BEV.

The uncertainty results for the MIMO-ID PointPillars and SECOND models trained on the KITTI dataset are shown in Tab. 4.10. While the MIMO-ID variant produces similar uncertainty estimation results as MIMO-BEV for PointPillars, it does not for SECOND, possibly because of capacity issues in the VFE of the latter. The TP and $\mathrm{FP}_{\mathrm{ML}}$ scores for classification are much higher for MIMO-ID (SC) and are outliers compared to the other models. The MIMO-ID (SC) model outputs predictions with much lower confidence compared to the other models. For the regression scores, the MIMO-ID (SC) model outperforms the PP model as expected based on lower regression uncertainty scores for SC models by comparing in Tab. 4.7 and Tab. 4.8. It also performs similarly to the other results for SECOND models.

Table 4.10: Uncertainty evaluation for the MIMO models trained over the KITTI dataset

Model	NLL (Cls) ↓			Brier Score ↓			NLL (Reg) ↓		Energy Score ↓		MCE	CE
	TP	$\mathrm{FP}_{\mathrm{ML}}$	$\mathrm{FP}_{\mathrm{BG}}$	TP	$\mathrm{FP}_{\mathrm{ML}}$	$\mathrm{FP}_{\mathrm{BG}}$	TP	$\mathrm{FP}_{\mathrm{ML}}$	TP	$\mathrm{FP}_{\mathrm{ML}}$	(Cls) ↓	(Reg) ↓
MIMO-BEV(PP)	**0.365**	0.445	0.987	**0.201**	0.261	0.763	-5.002	-2.340	0.433	0.670	0.207	0.064
MIMO-ID(PP)	0.376	**0.439**	1.004	0.212	**0.257**	0.772	-5.056	**-2.363**	0.454	0.719	0.199	**0.063**
MIMO-BEV(SC)	0.419	0.569	0.729	0.257	0.369	0.540	-4.311	-0.117	0.369	0.867	**0.185**	0.065
MIMO-ID(SC)	0.805	1.048	**0.472**	0.602	0.715	**0.302**	**-5.541**	-2.065	**0.299**	**0.599**	0.188	0.063

Fig. 4.3 visualizes qualitative results for the PointPillars models trained on the KITTI dataset. The close TP prediction has a high softmax score, lower Shannon entropy, and total variances. The further TP prediction has a lower softmax score, and increased Shannon entropy as well as aleatoric total variance. The mutual information is zero or close to zero for each prediction, while the epistemic total variance is also low but increases with distance. Epistemic total variance is the lowest for the MIMO-BEV model. Our results agree with previous work in this area displaying a large increase in aleatoric variance with distance away from the LiDAR sensor.

In Tab. 4.11 we compare different disagreement metrics for the multi-output models with 2 outputs and minimum cluster sizes of 2 and 1. Across all metrics, the MIMO-BEV model has lower disagreement signifying that the network capacity is too low to support fully independent subnetworks. The values are much lower than the original MIMO paper which had a disagreement of 3.2% for ensemble and MIMO compared to our results showing 0.06% disagreement. With a mimimum cluster size of 1, the disagreement is much larger due to clusters of size 1. These clusters shown in the % Cluster Size 1 column contribute

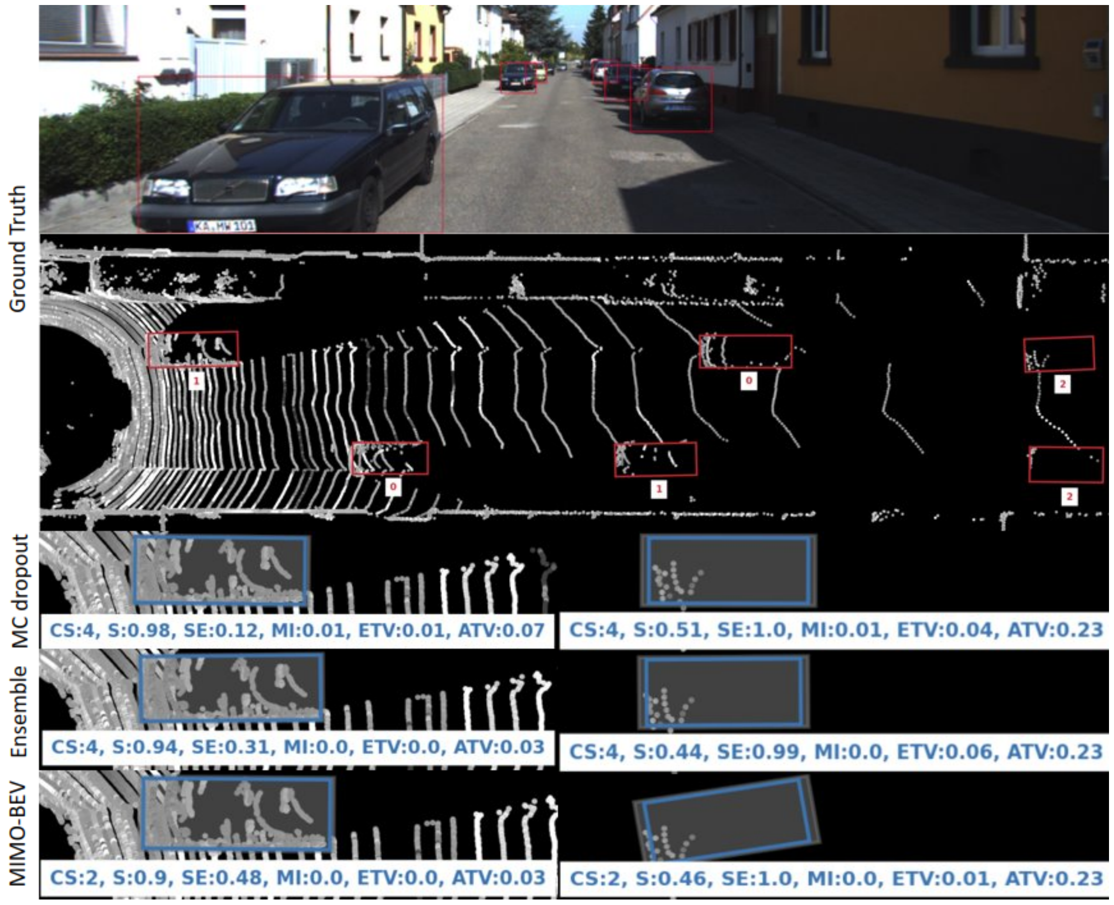
Ground Truth
MC dropout
Ensemble
MIMO-BEV
CS:4, S:0.98, SE:0.12, MI:0.01, ETV:0.01, ATV:0.07
CS:4, S:0.51, SE:1.0, MI:0.01, ETV:0.04, ATV:0.23
CS:4, S:0.94, SE:0.31, MI:0.0, ETV:0.0, ATV:0.03
CS:4, S:0.44, SE:0.99, MI:0.0, ETV:0.06, ATV:0.23
CS:2, S:0.9, SE:0.48, MI:0.0, ETV:0.0, ATV:0.03
CS:2, S:0.46, SE:1.0, MI:0.0, ETV:0.01, ATV:0.23

Figure 4.3: Top: Ground truth objects are displayed for this KITTI frame on the camera image as well as the BEV projection along with the difficulty number. Bottom: Each row is a multi-output model with two TP predictions. The left one is the truncated car in the camera view and the right one is the occluded yellow car in the distance. The blue box is the mean cluster prediction while the grey box is the 95% confidence interval for the width and height of the prediction. Details on cluster size, softmax score, Shannon entropy, mutual information, epistemic total variance and aleatoric total variance are included at the bottom.

almost all the disagreement. The D_{KL} is also increased significantly compared to minimum cluster size 2 due to the softmax distribution that we created for the calculation.

Table 4.11: Disagreement evaluation for the PointPillars models trained over the KITTI dataset

Model	Min Cluster 2					Min cluster 1		
	D	D_{KL}	D_{pos}	D_{diag}	D_{angle}	D	D_{KL}	% CS 1
MC dropout	0.0001	0.0157	0.1647	0.0697	0.6235	0.5672	1.2888	0.5672
Ensemble	0.0006	0.0403	0.1212	0.0855	0.7533	0.6198	1.4340	0.6196
MIMO-BEV	0.0000	0.0005	0.0121	0.0073	0.0519	0.0820	0.1814	0.0820

4.4.4 The ablation study on batch size and input repetition for MIMO-BEV

Tab. 4.12 and Tab. 4.13 contain the results for MIMO-BEV models trained for different batch sizes and an input repetition of 0.0 With increasing the batch size (BS) the mAP decreases due to the increase of frame groupings (pairings) without matching frames. Note that frame pairings during training are produced by sampling a batch, then duplicating it and shuffling. Using this procedure with a batch size of 3 creates a higher probability of matching frames than for batch size of 12. Due to the limited network capacity, fewer matching groupings result in a lower mAP. There are also fewer $\mathrm{FP}_{\mathrm{BG}}$ detections with a BS of 3 although the models with higher BSs have similar counts of $\mathrm{FP}_{\mathrm{BG}}$. The scoring rules are best for the batch size of 9, although the scores are comparable.

Table 4.12: The mAP and detection partitions for the PointPillars MIMO-BEV models trained over the KITTI dataset with increasing batch size (BS) and input repetition (IR) of 0.

BS	mAP (%) ↑	TP	$\mathrm{FP}_{\mathrm{ML}}$	$\mathrm{FP}_{\mathrm{BG}}$
		Counts Per Partition		
3	**64.61**	**8332**	**3857**	**56988**
6	64.01	8296	3904	61573
9	62.96	8243	3921	61571
12	61.86	8203	3928	59225

Tab. 4.14 and Tab. 4.15 contain the results for MIMO-BEV models trained with a batch size of 12 and increasing input repetitions of 0.0, 0.05, 0.1 and 0.2. A higher input

Table 4.13: Uncertainty evaluation for the PointPillars MIMO-BEV models trained on the KITTI dataset with increasing batch size (BS).

BS	NLL (Cls) ↓			Brier Score ↓			NLL (Reg) ↓		Energy Score ↓		MCE	CE
	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	TP	FP_{ML}	(Cls) ↓	(Reg) ↓
3	0.365	0.445	0.987	0.201	0.261	0.763	-5.002	-2.340	**0.433**	0.670	0.207	**0.064**
6	0.343	0.419	**0.983**	0.186	0.240	**0.763**	-4.884	**-2.397**	0.445	**0.664**	**0.190**	0.073
9	**0.343**	**0.418**	0.994	**0.186**	**0.237**	0.765	**-5.047**	-1.868	0.460	0.695	0.198	0.074
12	0.371	0.438	1.015	0.207	0.255	0.780	-4.909	-1.986	0.446	0.710	0.199	0.075

repetition increases the percentage of frame groupings that contain the same frame. As already explained, with the larger batch size, the mAP is lowered due to the increased randomness of these frame groupings. This randomness is offset by having higher input repetition, however. Do to the limited capacity of the detectors, matching frame groupings need to be increased to improve mAP, but at the cost of subnetwork independence.

Table 4.14: The mAP and detect partitions for the PointPillars MIMO-BEV models trained over the KITTI dataset with a batch size of 12 and input repetition (IR). The bold indicates the best performing model for the associated metric.

IR	mAP	TP	FP_{ML}	FP_{BG}
	(%) ↑	Counts Per Partition		
0.0	61.86	8203	3928	59225
0.05	63.2	8234	4000	57918
0.1	63.87	8250	3953	56968
0.2	**64.63**	**8297**	**3898**	**56279**

Table 4.15: Uncertainty evaluation for the PointPillars MIMO-BEV models trained on the KITTI dataset with a batch size of 12 and increasing input repetition (IR).

IR	NLL (Cls) ↓			Brier Score ↓			NLL (Reg) ↓		Energy Score ↓		MCE	CE
	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	TP	FP_{ML}	(Cls) ↓	(Reg) ↓
0.00	0.371	0.438	1.015	0.207	0.255	0.780	**-4.909**	-1.986	0.446	0.710	0.199	0.075
0.05	**0.361**	0.425	0.991	**0.199**	0.244	0.768	-4.815	-2.092	0.474	0.703	0.200	0.076
0.10	0.365	**0.419**	1.029	0.203	**0.242**	0.789	-4.804	**-2.233**	**0.465**	**0.689**	**0.182**	0.072
0.20	0.367	0.424	**0.988**	0.202	0.245	**0.764**	-4.697	-2.166	0.471	0.701	0.208	**0.068**

4.4.5 The ablation study on detection heads and network capacity for MIMO-BEV

To determine the effect of an increased number of detection heads on the network, we trained four MIMO-BEV models with a batch size of 3 and input repetition of 0. For each model, we vary the number of detection heads being two or three as well as the regular and wide versions of the backbone. We also train with our standard 120 epochs and an increased 150 epochs for the wide models. The wide version is created by doubling the number of filters for each VFE network and backbone component. By using the wide backbone, the network should be able to overcome capacity constraints and train with more detection heads [18].

Tab. 4.16 contains the average precision and detection results for the models. Our results show that there is a decrease of 0.5% mAP when adding an extra head to the network. This shows that the extra capacity of our network is not high enough to handle more detection heads. It also shows that switching the network to have the wide backbone causes a decrease in mAP of 1%. Surprisingly, the wide models are unable to achieve better performance for mAP, although they have a lower amount of FP_{BG} detections.

Table 4.16: The mAP and detect partitions for the PointPillars MIMO-BEV models trained over the KITTI dataset with increased detection heads as well as network capacity. The model type is specified by the number of heads along with a W for the WIDE models that have extra network capacity.

Type	Epochs	mAP (%) ↑	TP	FP_{ML}	FP_{BG}
			Counts Per Partition		
2	120	**64.61**	**8332**	3857	56988
2-W	120	63.57	8303	**3796**	52159
2-W	150	64.15	8262	3931	**51705**
3	120	64.14	8300	3969	72885
3-W	120	63.07	8295	3866	58678
3-W	150	63.37	8269	3925	54392

Tab. 4.17 contains the results for the scoring rules and calibration errors for our models. In these results we can see that the wide models perform better across all metrics compared to the regular models. The 2 head models perform better for regression metrics, while the wide models with extra epochs performed best for classification. Thus, adding either extra capacity or an extra head to the regular variant slightly improved the uncertainty results, i.e., lower scores.

Table 4.17: Uncertainty evaluation for the PointPillars MIMO-BEV models trained on the KITTI dataset with increased detection heads as well as network capacity. The model type is specified by the number of heads along with a W for the WIDE models that have extra network capacity.

Type	Epochs	NLL (Cls) $\downarrow$			Brier Score $\downarrow$			NLL (Reg) $\downarrow$		Energy Score $\downarrow$		MCE	CE
		TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	FP_{BG}	TP	FP_{ML}	TP	FP_{ML}	(Cls) $\downarrow$	(Reg) $\downarrow$
2	120	0.365	0.445	0.987	0.201	0.261	0.763	-5.002	-2.340	0.433	0.670	0.207	0.064
2-W	120	0.359	0.417	1.015	0.195	0.240	0.781	**-5.064**	**-2.551**	0.451	**0.656**	0.204	**0.058**
2-W	150	0.359	**0.414**	1.037	0.196	**0.236**	0.797	-5.020	-0.925	**0.403**	0.722	0.210	0.061
3	120	0.359	0.460	0.944	0.200	0.271	0.728	-4.833	-1.952	0.472	0.704	0.186	0.073
3-W	120	0.373	0.434	0.983	0.207	0.251	0.760	-4.990	-2.184	0.406	0.667	0.201	0.059
3-W	150	**0.350**	0.452	**0.943**	**0.193**	0.267	**0.724**	-5.026	-2.131	0.411	0.702	**0.185**	**0.058**

Chapter 5

Conclusion

LiDAR-MIMO offers fast and high-quality detection and uncertainty estimation, which is crucial for safety-critical and resource-constrained robotic vision, such as found in autonomous driving. In our experiments, LiDAR-MIMO achieves consistently higher mAP than MC dropout, while approaching ensembles. Further, similar to the other two methods, LiDAR-MIMO produces high-quality uncertainty estimates as measured by calibration errors and scoring rules, but without the large runtime overhead of the other methods. Our design for adapting MIMO to LiDAR-based 3D object detection relies on the composition of BEV feature maps, which is applicable to efficient 3D object detectors, such as PointPillars and SECOND. The stacking approach is likely to work well for other LiDAR-based 3D object detectors that are based on LiDAR BEV projections, but also LiDAR range images. Further, MIMO-ID can be used for any LiDAR-based detector, but it comes with increased data processing time. In general, MIMO is limited by the capacity of the underlying detector, which in our experiments allowed us to fit only two sub-networks. In future work, we plan to explore LiDAR-MIMO with larger capacity networks and more detection heads, and adapting LiDAR-MIMO to other 3D detector designs, such as ones based on LiDAR range images.

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[2] Glenn W Brier. Verification of Forecasts Expressed in Terms of Probability. *Monthly weather review*, 78(1):1–3, 1950.

[3] 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.

[4] Ángela Casado-García and Jónathan Heras. Ensemble Methods for Object Detection. In *ECAI 2020*, pages 2688–2695. IOS Press, 2020.

[5] Sebastian Cygert and Andrzej Czyzewski. Robust object detection with multi-input multi-output faster r-cnn. *arXiv preprint arXiv:2111.13065*, 2021.

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[7] 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, 2018. doi: 10.1109/ITSC.2018.8569814.

[8] Di Feng, Lars Rosenbaum, Claudius Glaeser, Fabian Timm, and Klaus Dietmayer. Can We Trust You? On Calibration of a Probabilistic Object Detector for Autonomous Driving. *arXiv preprint arXiv:1909.12358*, 2019.

[9] Di Feng, Ali Harakeh, Steven L Waslander, and Klaus Dietmayer. A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[10] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[11] 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, 2016.

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[13] Tilmann Gneiting and Adrian E Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[14] Tilmann Gneiting, Larissa I Stanberry, Eric P Grimit, Leonhard Held, and Nicholas A Johnson. Assessing probabilistic forecasts of multivariate quantities, with an application to ensemble predictions of surface winds. *Test*, 17(2):211, 2008.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[16] 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.

[17] Ali Harakeh and Steven L. Waslander. Estimating and Evaluating Regression Predictive Uncertainty in Deep Object Detectors. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YLewtnvKgR7.

[18] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OGg9XnKxFAH.

[19] Dan Hendrycks and Kevin Gimpel. A baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *Proceedings of International Conference on Learning Representations*, 2017.

[20] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing Error in Object Detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012.

[21] Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7141-what-uncertainties-do-we-need-in-bayesian-deep-learning-for-computer-vision.pdf`.

[22] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[23] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate Uncertainties for Deep Learning Using Calibrated Regression. In *International Conference on Machine Learning*, pages 2796–2804. PMLR, 2018.

[24] Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified Uncertainty Calibration. *Advances in Neural Information Processing Systems*, 32:3792–3803, 2019.

[25] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[26] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[28] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[29] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. *Advances in Neural Information Processing Systems*, 32:13153–13164, 2019.

[30] Dimity Miller, Niko Sünderhauf, Haoyang Zhang, David Hall, and Feras Dayoub. Benchmarking Sampling-based Probabilistic Object Detectors. In *CVPR Workshops*, volume 3, page 6, 2019.

[31] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining Well Calibrated Probabilities Using Bayesian Binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[32] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[33] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[34] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian Adverse Driving Conditions Dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, 2021.

[35] 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.

[36] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[37] Rui Qian, Xin Lai, and Xirong Li. 3D Object Detection for Autonomous Driving: A Survey. *arXiv preprint arXiv:2106.10823*, 2021.

[38] Felippe Schmoeller Roza, Maximilian Henne, Karsten Roscher, and Stephan Günnemann. Assessing Box Merging Strategies and Uncertainty Estimation Methods in Multimodel Object Detection. In *European Conference on Computer Vision*, pages 3–10. Springer, 2020.

[39] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[40] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[41] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[43] OpenPCDet Development Team. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[44] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, 2018.

[45] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3D Object Detection from Point Clouds. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[46] Yuanxin Zhong, Minghan Zhu, and Huei Peng. Uncertainty-Aware Voxel based 3D Object Detection and Tracking with von-Mises Loss, 2020.

[47] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.