LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving

Gregory P. Meyer*, Ankit Laddha*, Eric Kee, Carlos Vallespi-Gonzalez, Carl K. Wellington
Uber Advanced Technologies Group
{gmeyer,aladdha,ekee,cvallespi,cwellington}@uber.com

Abstract

In this paper, we present LaserNet, a computationally efficient method for 3D object detection from LiDAR data for autonomous driving. The efficiency results from processing LiDAR data in the native range view of the sensor, where the input data is naturally compact. Operating in the range view involves well known challenges for learning, including occlusion and scale variation, but it also provides contextual information based on how the sensor data was captured. Our approach uses a fully convolutional network to predict a multimodal distribution over 3D boxes for each point and then it efficiently fuses these distributions to generate a prediction for each object. Experiments show that modeling each detection as a distribution rather than a single deterministic box leads to better overall detection performance. Benchmark results show that this approach has significantly lower runtime than other recent detectors and that it achieves state-of-the-art performance when compared on a large dataset that has enough data to overcome the challenges of training on the range view.

1. Introduction

3D object detection is a key capability for autonomous driving. LiDAR range sensors are commonly used for this task because they generate accurate range measurements of the objects of interest independent of lighting conditions. To be used in a real-time autonomous system, it is important that these approaches run efficiently in addition to having high accuracy. Also, within the context of a full self-driving system, it is beneficial to have an understanding of the detector's uncertainty.

LiDAR range sensors used on autonomous vehicles employ a variety of physical approaches to point one or more range-measuring lasers in different directions (physically spinning, raster scanning, MEMS mirrors, etc), but the data from these sensors are all similar in that they contain range measurements from a single location spanning some verti-
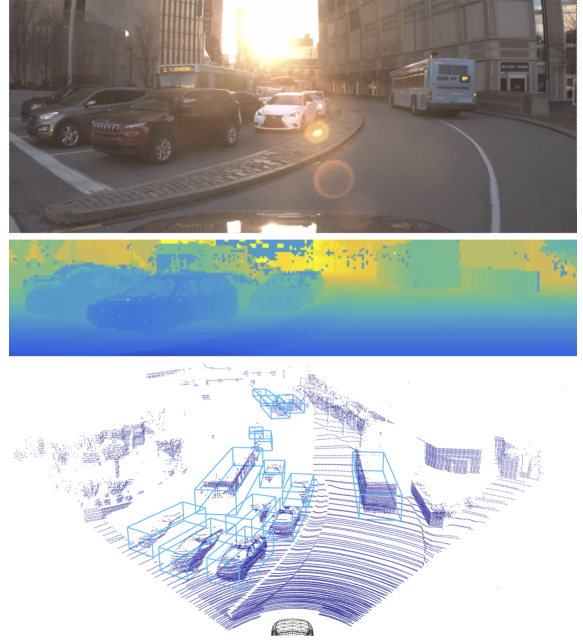
*Equal Contribution

Figure 1: Example detection results from our approach (bottom) generated from the input range image (middle). The camera image (top) is provided for reference. Notice the data is dense in the range image but becomes sparse when projected into 3D space.

cal and horizontal angular field of view. Note that this data is quite different from full 3D point cloud data collected by scanning an object on a turntable or through other means to generate a consistent dense point cloud of all sides of the object. As shown in Figure 1, the LiDAR data is inherently dense from the sensor's point of view but sparse when the range measurements are projected into 3D space. The density of measurements is constant in angle, so nearby objects have significantly more measurements than far away objects. Also, only the sides of objects facing the sensor receive measurements.

When performing 2D detection in camera images, efficient and high-performing results have been achieved using dense convolutional approaches [19, 24]. Although the sen-

sor's range data is similar in many ways to a camera image, the desired output is an oriented bounding box in 3D space. Therefore, 3D object detectors must transform the sensor data in the range view (RV) into detections in the top-down view also known as the bird's eye view (BEV). Most existing work starts by projecting the range measurements into 3D points and then discretizing the 3D space into voxels in order to operate directly in the output space [27, 28, 17]. These approaches have shown impressive results but require operating on sparse input data which can be computationally expensive. There has also been work operating on the dense RV representation directly [5, 16], but these methods have not matched the performance of the BEV approaches. Additionally, there has been work combining both RV and BEV representations [5].

These representations each have different advantages and disadvantages. In the RV, the sensor data is dense, but the perceived size of objects varies with range. In the BEV, the data is sparse, but the size of objects remains constant regardless of range. This consistency adds a strong prior to the predictions making the problem easier to learn. Lastly, the RV preserves occlusion information which is lost when projecting the data into the BEV.

In this work, we present an efficient method for learning a probabilistic 3D object detector in an end-to-end fashion. When there is sufficient training data, we achieve state-of-the-art detection performance with a significantly lower runtime. Our approach is efficient because we use a small and dense range image instead of a large and sparse bird's eye view image. Our proposed method produces not only a class probability for each detection but also a probability distribution over detection bounding boxes. To our knowledge, our proposed method is the first to capture the uncertainty of a detection by modeling the distribution of bounding box corners. By estimating the accuracy of a detection, our approach enables downstream components in a full self-driving system to behave differently around objects with varying levels of uncertainty.

2. Related Work

2.1. 3D Object Detection

Multiple deep learning based methods have been proposed for 3D object detection for autonomous driving. VeloFCN [16] and MV3D [5] proposed methods using a range view representation. Both methods construct the RV by discretizing the azimuth and elevation angles. In this paper, we propose a different RV representation using the configuration of the LiDAR, and demonstrate that it leads to better performance.

VoxelNet [30] computes PointNet [23] style features from the LiDAR points for each 3D voxel. Afterwards, they run 3D convolutions over these voxels to produce de-

tections. Instead of voxelizing the input, our approach operates directly on the raw range data and aggregates predictions from LiDAR points through mean shift clustering to generate the detections.

Similar to 2D image detection, 3D methods can be grouped into two meta frameworks: region proposal methods [24] and single shot methods [19]. Region proposal based methods [5, 15, 22] have two stages: the first stage proposes plausible regions containing objects, and the second stage extracts features from the proposed regions and uses them to produce detections. Single shot based methods [1, 16, 28, 30] produce detections with a single stage network. In this work, we use a single shot approach since region proposal networks are often computationally expensive and minimizing latency is critical for a real-time autonomous system.

In addition to using LiDAR data, several previous works use images [5, 15, 22, 17, 26, 21, 9] or high definition maps [27] to improve detections. In this paper, we only utilize LiDAR data and leave sensor fusion as future work. Some approaches [4, 3, 20, 25] have tried to tackle 3D detection without using range data from LiDAR. However, the accuracy of these methods do not reach the performance of LiDAR based methods.

2.2. Probabilistic Object Detection

Most of the state-of-the-art 2D [24, 19] and 3D [5, 15, 27, 17] object detection methods produce a single box with a probability score for each detection. While this probability encompasses the existence and semantic uncertainty, it does not accurately measure the localization uncertainty. Recently, [12] proposed to explicitly predict the intersection-over-union (IoU) of each 2D detection with the ground truth label. They use the predicted IoU to refine and score the detections. Estimating the IoU provides a measure of the detection's localization error. In this paper, we estimate a probability distribution over box detections instead of only the mean, and the variance of the predicted distribution indicates the amount of uncertainty in the position of the box corners.

The seminal work in [13] studies Bayesian deep networks for multiple computer vision tasks. Based on this work, [6, 7] produce and analyze epistemic and aleatoric uncertainties in 3D object detection. However, they do not use the predicted uncertainties to significantly improve the performance of the detector. In this paper, we focus on estimating the aleatoric uncertainty by way of predicting a probability distribution over box detections, and we leverage the predicted distribution to combine and refine our detections. The epistemic uncertainty is not estimated by our method since it currently cannot be computed efficiently.
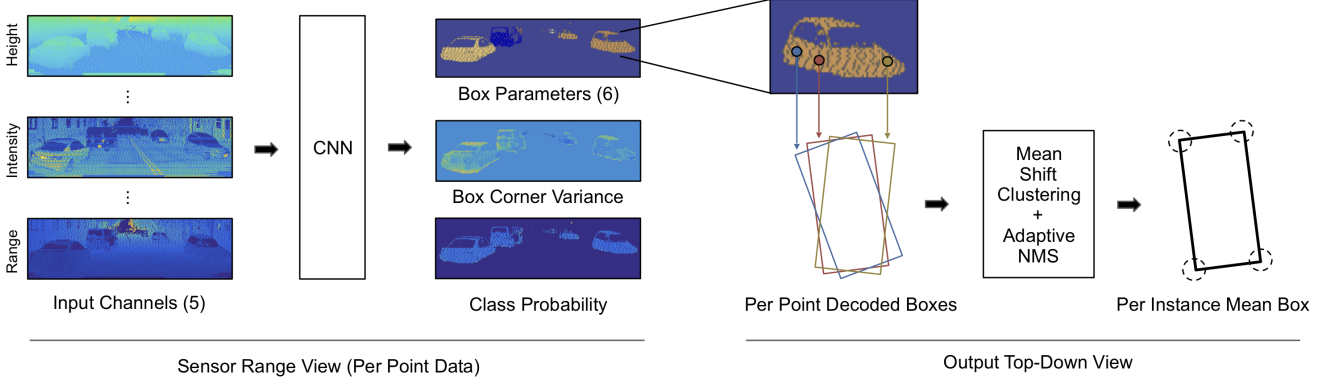
Figure 2: An overview of our approach to 3D object detection. We use the inherent range view representation of the sensor to build a dense input image (Section 3.1). The image is passed through a fully-convolutional network to produce a set of predictions (Section 3.2). For each LiDAR point in the image, we predict a class probability, and we regress a probability distribution over bounding boxes in the top-down view (Section 3.3). These per-point distributions are combined through mean shift clustering to reduce the noise in the individual predictions (Section 3.4). The entire detector is trained end-to-end with the loss defined on the box corners (Section 3.5). At inference, we leverage a novel adaptive non-maximum suppression (NMS) algorithm to remove duplicate box distributions (Section 3.6).

3. Proposed Method

An overview of our proposed method is shown in Figure 2. In the following sections each step is described in detail.

3.1. Input Representation

The LiDAR produces a cylindrical range image as it sweeps over the environment with a set of lasers. The horizontal resolution of the image is determined by the rotation speed and laser pulse rate, and the vertical resolution is determined by the number of lasers. The Velodyne 64E LiDAR contains a set of 64 lasers with a non-uniform vertical spacing of approximately $0.4°$ and has a horizontal angular resolution of approximately $0.2°$. For each point in the sweep, the sensor provides a range r, reflectance e, azimuth θ, and laser id m, which corresponds to a known elevation angle. Using the range, azimuth, and elevation, we can compute the corresponding 3D point (x, y, z) in the sensor frame. We build an input image by directly mapping the laser id m to rows and discretizing azimuth θ into columns. If more than one point occupies the same cell in the image, we keep the closest point.

For each cell coordinate in the image, we collect a set of input channels from its corresponding point: range r, height z, azimuth angle θ, intensity e, and a flag indicating whether the cell contains a point. The result is a five channel image that forms the input to our network (see Figure 2).

3.2. Network Architecture

Our image contains objects at a wide range of distances, and the size of an object can vary from several thousand points to a single point. We leverage the deep layer aggregation network architecture [29] to effectively extract and combine multi-scale features. Our network is fully-convolutional and consists of three hierarchical levels as shown in Figure 3. The size of convolutional kernels at each level is 64, 64, 128 respectively.

Each level contains a feature extraction module and some number of feature aggregation modules. The structure of these modules is depicted in Figure 4. Since the horizontal resolution of the image is significantly larger than the vertical resolution, we keep the vertical resolution constant and only perform downsampling and upsampling along the horizontal dimension. A final 1×1 convolutional layer is used to transform the resulting feature map to our encoded predictions.

3.3. Predictions

Our network is trained to predict a set of class probabilities for each point in the image. Given a point lies on an object, the network predicts a probability distribution over bounding boxes that could encapsulate the object. The probability distribution can be multimodal when the observed data is incomplete, which often occurs at long range, when points are sparse, and when there are occlusions. Therefore, we model the probability distribution with a mixture model, and train the network to predict a set of means with corresponding variances and mixture weights. The distribution learned by the network is defined by the loss function, which is discussed in Section 3.5.

For the application of autonomous driving, we assume all objects lie on the same ground plane; therefore, we can characterize a bounding box by its four corners in the x-y plane. The 3D bounding boxes depicted in Figure 1 are pro-
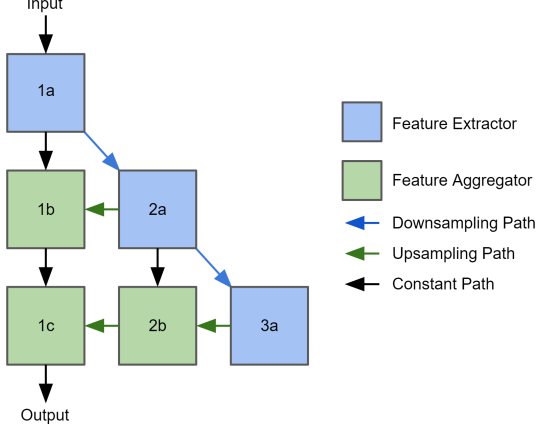
Figure 3: Our deep layer aggregation network architecture. The columns indicate different resolution levels, and the rows indicate aggregation stages.
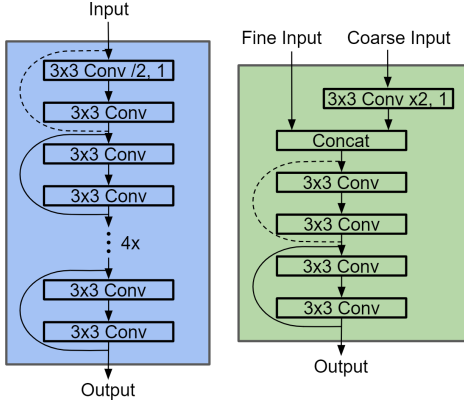

Figure 4: Our feature extraction module (left) and feature aggregation module (right). The modules are residual networks [11]. Dashed lines indicate a convolution is performed to reshape the feature map.

duced using an assumed ground plane and a fixed height. Instead of regressing the four corners directly, we predict a relative center (d_x, d_y), a relative orientation $(\omega_x, \omega_y) = (\cos\omega, \sin\omega)$, and the dimensions (l, w) of the box. We compute the box's absolute center $\boldsymbol{b}_c$ and absolute orientation ϕ as follows:

$$\boldsymbol{b}_c = [x, y]^T + \boldsymbol{R}_\theta [d_x, d_y]^T$$
$$\phi = \theta + \operatorname{atan2}(\omega_y, \omega_x) \tag{1}$$

where (x, y) and θ is the 2D position and azimuth angle of the LiDAR point, and $\boldsymbol{R}_\theta$ is the rotation matrix parameterized by θ. Subsequently, we can calculate the four corners

of the bounding box,

$$\boldsymbol{b}_1 = \boldsymbol{b}_c + \frac{1}{2}\boldsymbol{R}_\phi [l, w]^T \qquad \boldsymbol{b}_2 = \boldsymbol{b}_c + \frac{1}{2}\boldsymbol{R}_\phi [l, -w]^T$$
$$\boldsymbol{b}_3 = \boldsymbol{b}_c + \frac{1}{2}\boldsymbol{R}_\phi [-l, -w]^T \quad \boldsymbol{b}_4 = \boldsymbol{b}_c + \frac{1}{2}\boldsymbol{R}_\phi [-l, w]^T \tag{2}$$

where $\boldsymbol{R}_\phi$ is the rotation matrix parameterized by ϕ. For convenience, we concatenate the four corners into a single vector, $\boldsymbol{b} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_4]$.

To simplify the predict probability distribution, we assume a uniform variance across both the x and y dimensions, and we share the variance across all four corners of the bounding box. As in [13], we train the network to predict the log of the standard deviation $s = \log\sigma$.

Altogether, for each point in the image, our network predicts a set of class probabilities $\{p_c\}_{c=1}^{C}$, where C is the number of object classes in addition to a background class. For each point and class of object, the network predicts a set of bounding box parameters $\{d_{x,k}, d_{y,k}, \omega_{x,k}, \omega_{y,k}, l_k, w_k\}_{k=1}^{K}$, a set of log standard deviations $\{s_k\}_{k=1}^{K}$, and a set of mixture weights $\{\alpha_k\}_{k=1}^{K}$, where K is the number of components in the mixture model.

3.4. Mean Shift Clustering

Each point independently predicts the distribution of bounding boxes; however, points on the same object should predict a similar distribution. Naturally, the individual predictions will contain some amount of noise. We can reduce this noise by combining the per-point predictions through mean shift clustering. Since our predicted distribution is class dependent and multimodal, we perform mean shift on each object class and each component of the mixture model separately. For efficiency, mean shift is performed over box centers instead of box corners, which reduces the dimensionality of the problem. Additionally, we propose a fast approximation of the mean shift algorithm when operating in two dimensions.

Our approximate algorithm begins by discretizing the top-down view into bins of size Δx by Δy. For each bin i, which contains one or more box centers, we create an initial mean $\boldsymbol{m}_i$ by averaging over all the centers in that bin. In addition, we record the set of points S_i whose box centers fall inside the bin. We iteratively update the means as follows:

$$\boldsymbol{m}_i \leftarrow \frac{\sum_{j \in i \cup N(i)} K_{i,j}(\boldsymbol{m}_j \cdot |S_j|)}{\sum_{j \in i \cup N(i)} K_{i,j}|S_j|} \tag{3}$$

where

$$K_{i,j} = \exp\left(-\frac{\|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2}{\Delta x^2 + \Delta y^2}\right) \tag{4}$$

and $N(i)$ is the set of eight bins neighboring the ith bin. After the update, if the ith mean now falls into the jth bin,

we merge i into j,

$$\boldsymbol{m}_j \leftarrow \frac{\boldsymbol{m}_i \cdot |S_i| + \boldsymbol{m}_j \cdot |S_j|}{|S_i| + |S_j|} \quad S_j \leftarrow S_i \cup S_j \quad (5)$$

and invalidate $\boldsymbol{m}_i \leftarrow \boldsymbol{0}$ and $S_i \leftarrow \emptyset$.

Equation (3) can be computed efficiently by exploiting the regular structure of the bins. By constructing a tensor of bins and generating shifted versions of this tensor, we can update all the means simultaneously using only element-wise operators. This type of computation is well suited for a graphics processing unit (GPU).

After performing a fixed number of mean shift iterations, per-point box distributions that are assigned to the same cluster are combined. Specifically, we model the cluster bounding box distribution as the product of the per-point distributions. The mean and variance of the cluster probability distribution is

$$\hat{\boldsymbol{b}}_i = \frac{\sum_{j \in S_i} w_j \boldsymbol{b}_j}{\sum_{j \in S_i} w_j} \quad \hat{\sigma}_i^2 = \left(\sum_{j \in S_i} \frac{1}{\sigma_j^2} \right)^{-1} \quad (6)$$

where $w = 1/\sigma^2$. Each point's predicted bounding box and variance is replaced by the bounding box and variance of its cluster. For our experiments, we perform three iterations of mean shift with $\Delta x = \Delta y = 0.5$ meters.

3.5. End-to-end Training

Mapping from box parameters to box corners, equations (1) and (2), and merging the bounding boxes distributions, equation (6), are differentiable; therefore, we are able to train the network in an end-to-end fashion.

For each point in the image, we use the multi-class cross entropy loss $\mathcal{L}_{\text{prob}}$ to learn the class probabilities $\{p_c\}_{c=1}^C$. To handle class imbalance, we employ focal loss [18], which is a modified version of the cross entropy loss. The classification loss for the entire image is defined as follows:

$$\mathcal{L}_{\text{cls}} = \frac{1}{P} \sum_i \mathcal{L}_{\text{prob},i} \quad (7)$$

where $\mathcal{L}_{\text{prob},i}$ is the loss for the ith point in the image, and P is the total number of points in the image.

For each point on an object, we learn the parameters of the object's mixture model by first identifying which component best matches the ground truth,

$$k^* = \arg\min_k \|\hat{\boldsymbol{b}}_k - \boldsymbol{b}^{\text{gt}}\| \quad (8)$$

where $\hat{\boldsymbol{b}}_k$ is the kth mean component of the mixture model and $\boldsymbol{b}^{\text{gt}}$ is the corresponding ground truth bounding box. Afterwards, we update the parameters of the k^* component following the approach proposed in [13],

$$\mathcal{L}_{\text{box}} = \sum_n \frac{1}{\hat{\sigma}_{k^*}} \left| \hat{b}_{n,k^*} - b_n^{\text{gt}} \right| + \log \hat{\sigma}_{k^*} \quad (9)$$

where $\hat{b}_{n,k^*}$ is the nth element of $\hat{\boldsymbol{b}}_{k^*}$, and b_n^{gt} is the corresponding ground truth value. As discussed in [13], this loss imposes a Laplacian prior on the learned distribution. Next, we update the mixture weights $\{\alpha_k\}_{k=1}^K$ again using the multi-class cross entropy loss $\mathcal{L}_{\text{mix}}$, where the positive label corresponds to the k^* component. The idea of only updating the prediction that best matches the ground truth was originally proposed in [10] as the hindsight loss for multiple choice learning. The regression loss for the entire image is defined as follows:

$$\mathcal{L}_{\text{reg}} = \frac{1}{N} \sum_i \frac{\mathcal{L}_{\text{box},i} + \lambda \mathcal{L}_{\text{mix},i}}{n_i} \quad (10)$$

where $\mathcal{L}_{\text{box},i}$ and $\mathcal{L}_{\text{mix},i}$ are the losses for the ith point in the image which is on an object, n_i is the total number of points that lie on the same object as i, N is the total instances of objects in the image, and λ is the relative weighting of the two losses. We set $\lambda = 0.25$ for all our experiments. The total loss for the image is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{reg}}$.

3.6. Adaptive Non-Maximum Suppression

At inference, we identify the points that belong to an object class by thresholding the predicted class probability p_c, and we use a threshold of $1/C$ in all our experiments. As described previously, each point on an object predicts a probability distribution over bounding boxes. For each predicted distribution, we extract a set of bounding boxes which correspond to the means of the mixture model.

Typically non-maximum suppression (NMS) is performed to remove redundant bounding boxes. The standard procedure is to identify boxes with an intersection-over-union (IoU) greater than a fixed threshold and remove the box with the lower class probability. This strategy is inappropriate for our method for two reasons. First, it does not consider the predicted variance of the bounding boxes. Second, the class probability in our case does not indicate the quality of the bounding box. For example, it is relatively easy to classify the front of a semi-truck as a vehicle, but it is difficult to accurately predict its extents.

Alternatively, we propose an adaptive NMS algorithm, which uses the predicted variance to determine an appropriate IoU threshold for a pair of boxes. In addition, we utilize the likelihood of the box as its score. Since we use the means of the mixture model, the likelihood of the box corresponding to the kth component of the mixture model reduces to $\alpha_k / 2\hat{\sigma}_k$.

In the top-down view, bounding boxes should not overlap; however, due to the uncertainty in the predictions, some amount of overlap is expected. For each pair of overlapping bounding boxes from the same class, we calculate the upper-bound on the IoU given their predicted variances. We accomplish this by assuming the worst-case scenario, which
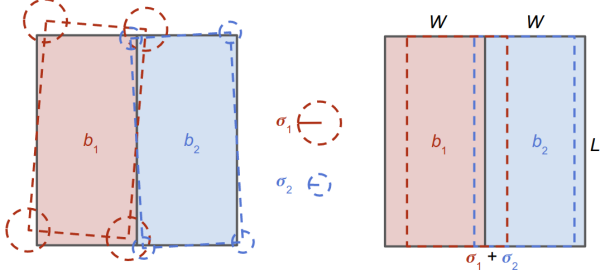
Figure 5: An illustration of our adaptive NMS technique. Consider a pair of vehicles positioned side-by-side. The dashed outlines on the left depict a possible set of predictions produced by our method. To determine whether the bounding boxes encapsulate unique objects, we utilize the predicted variances (shown in the center) to estimate the worst-case overlap as shown on the right. In this example, both bounding boxes would be preserved because the actual overlap is less than the estimated worst-case overlap.

occurs when two objects of the same dimensions are side-by-side, as shown in Figure 5. In this case, the maximum tolerated IoU t should be,

$$t = \begin{cases} \frac{\sigma_1 + \sigma_2}{2w - \sigma_1 - \sigma_2} & \sigma_1 + \sigma_2 < w \\ 1 & \text{otherwise} \end{cases} \qquad (11)$$

where w is the average width of the object's bounding box. For example, $w \approx 2$ meters for vehicles.

If the IoU of a pair of boxes exceeds the maximum threshold, then either one of the predicted boxes is incorrect or one of the predicted variances is incorrect. Assuming the bounding box is wrong, we can remove the box with the lower likelihood. Otherwise, we can assume the variance is inaccurate and increase the variance of the less likely box such that t would equal the observed IoU. The former is referred to as Hard NMS while the latter is analogous to Soft NMS [2]. The effect of this choice is examined in the ablation study in Section 4.2.

4. Experiments

Our proposed approach is evaluated and compared against state-of-the-art methods on two datasets: the large-scale ATG4D object detection dataset, and the small-scale KITTI object detection benchmark [8].

4.1. Evaluation on ATG4D

The ATG4D dataset contains 5,000 sequences for training and 500 for validation. The sequences from the train set are sampled at 10 Hz, while the validation set are sampled at 0.5 Hz. The entire dataset contains 1.2 million sweeps for training, and 5,969 sweeps for validation. All sweeps are captured using a Velodyne 64E LiDAR. We adopt the

same evaluation procedure as the KITTI benchmark and only consider the detections within the front 90° field of view of the sensor and up to 70 meters. As a result, our range image contains only this front portion of the LiDAR data and has dimensions of 512×64.

The network is trained for 300k iterations using the Adam optimizer [14] with a learning rate of 0.002 exponential decayed at a rate of 0.99 every 150 iterations. We use a batch size of 128 distributed over 32 GPUs. For vehicles, we learn a multimodal distribution with three components ($K = 3$), and for pedestrians and bikes, we learn a unimodal distribution ($K = 1$).

Table 1 shows our results on the validation set compared to other state-of-the-art methods. Like the KITTI benchmark, we compute the average precision (AP) at a 0.7 IoU for vehicles and a 0.5 IoU for bikes and pedestrians. On this dataset, our approach out-performs existing state-of-the-art methods in the $0-70$ meter range. Furthermore, our method out-performs the LiDAR-only methods at all ranges, and it is only surpassed by a LiDAR+RGB method on vehicles and bikes at long range where the additional image data provides the most value. We believe our approach performs significantly better on pedestrians because our range view representation does not discretize the input data into voxels.

4.2. Ablation Study on ATG4D

In this section, we examine the various aspects of our proposed method, and their effects on vehicle detection performance. We conduct the ablation study on the ATG4D dataset, and the results are shown in Table 2. A detailed description of the study is provided below.

Predicting a Probability Distribution
The largest improvement is a result of predicting a distribution of bounding boxes instead of merely the mean. When only predicting the mean bounding box, equation (6) becomes a simple average, and equation (9) reduces to the ℓ_1 loss on the box corners. Furthermore, the score of the bounding box is the class probability in this case. We believe the loss in performance is due to the probability not being well correlated with the accuracy of the box.

Image Formation
Previous methods which utilize the range view [5, 16] uniformly discretize the elevation angle into rows. However, the lasers in the Velodyne 64E LiDAR are not uniformly spaced. A gain in performance can be obtained by mapping points to rows using the laser id and processing the data directly as the sensor captured it.

Mean Shift Clustering
Without clustering, each point independently predicts a distribution of bounding boxes. These individual box predictions naturally contain some amount of noise, and we can

Table 1: BEV Object Detection Performance on ATG4D

Method	Input	Vehicle $AP_{0.7}$				Bike $AP_{0.5}$				Pedestrian $AP_{0.5}$			
		0-70m	0-30m	30-50m	50-70m	0-70m	0-30m	30-50m	50-70m	0-70m	0-30m	30-50m	50-70m
LaserNet (Ours)	LiDAR	**85.34**	**95.02**	**84.42**	67.65	**61.93**	**74.62**	51.37	40.95	**80.37**	**88.02**	**77.85**	**65.75**
PIXOR [28]	LiDAR	80.99	93.34	80.20	60.19	-	-	-	-	-	-	-	-
PIXOR++ [27]	LiDAR	82.63	93.80	82.34	63.42	-	-	-	-	-	-	-	-
ContFuse [17]	LiDAR	83.13	93.08	82.48	65.53	57.27	68.08	48.83	38.26	73.51	80.60	71.68	59.12
ContFuse [17]	LiDAR+RGB	85.17	93.86	84.41	**69.83**	61.13	72.01	**52.60**	**43.03**	76.84	82.97	75.54	64.19

Table 2: Ablation Study on ATG4D

Predicted Distribution	Image Spacing	Mean Shift	IoU Threshold	NMS Type	Vehicle $AP_{0.7}$
Mean-only	Laser	Yes	0.1	Hard	77.05
Unimodal	Uniform	Yes	0.1	Hard	79.14
Unimodal	Laser	No	0.1	Hard	80.22
Unimodal	Laser	Yes	0.1	Hard	80.92
Multimodal	Laser	Yes	0.1	Hard	81.80
Multimodal	Laser	Yes	N/A	Soft	84.43
Multimodal	Laser	Yes	Adaptive	Hard	83.68
Multimodal	Laser	Yes	Adaptive	Soft	**85.34**

Table 3: Runtime Performance on KITTI

Method	Forward Pass (ms)	Total (ms)
LaserNet (Ours)	**12**	**30**
PIXOR [28]	35	62
PIXOR++ [27]	35	62
VoxelNet [30]	190	225
MV3D [30]	-	360
AVOD [15]	80	100
F-PointNet [22]	-	170
ContFuse [17]	60	-

Table 4: BEV Object Detection Performance on KITTI

Method	Input	Vehicle $AP_{0.7}$		
		Easy	Moderate	Hard
LaserNet (Ours)	LiDAR	78.25	73.77	66.47
PIXOR [28]	LiDAR	81.70	77.05	72.95
PIXOR++ [27]	LiDAR	**89.38**	83.70	**77.97**
VoxelNet [30]	LiDAR	89.35	79.26	77.39
MV3D [5]	LiDAR+RGB	86.02	76.90	68.49
AVOD [15]	LiDAR+RGB	88.53	83.79	77.90
F-PointNet [22]	LiDAR+RGB	88.70	84.00	75.33
ContFuse [17]	LiDAR+RGB	88.81	**85.83**	77.33

reduce this noise by combining the independent predictions through mean shift clustering.

Non-Maximum Suppression
When the LiDAR points are sparse, there are multiple configurations of bounding boxes that could explain the observed data. By predicting a multimodal distribution at each point, we can further improve the recall of our method. It is inappropriate to use NMS with a strict threshold when producing a multimodal distribution because only the most likely bounding box will persist. Alternatively, we could use Soft NMS [2] to re-weight the confidence instead of eliminating boxes, but this breaks our probabilistic interpretation of the confidence. By leveraging our adaptive NMS algorithm, we maintain our probabilistic interpretation and obtain better performance.

4.3. Runtime Evaluation

Runtime performance is equally important for the purpose of autonomous driving. Table 3 compares the runtime performance between our approach (measured on a NVIDIA 1080Ti GPU) and existing methods on KITTI. The forward pass refers to the amount of time it takes to run the network, and the total includes pre and post processing in addition to the forward pass. Our proposed method is twice as fast as the fastest state-of-the-art method. We are able

to achieve a faster runtime because we operate on a small dense range view image instead of a large and sparse bird's eye view representation.

4.4. Evaluation on KITTI

The KITTI object detection benchmark [8] contains 7,481 training sweeps and 7,518 testing sweeps captured by a Velodyne 64E LiDAR. For the training set, object annotations are provided within the front $90°$ field of view of the LiDAR up to approximately 70 meters.
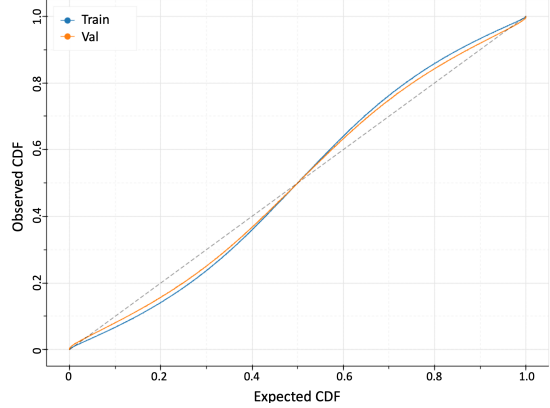
To train our network, we use 5,985 sweeps from the training set, and we holdout the remaining sweeps for validation. We train the network for 50k iterations with the same learning schedule as before, and we use a batch size of 12 on a single GPU. To help avoid overfitting on this small training set, we use data augmentation. We randomly flip the range image, and we apply random pixel shifts in the horizontal dimension.

Learning a probability distribution, especially a multimodal distribution, over bounding boxes proved to be difficult on such a small dataset. Therefore, we trained the network to only detect vehicles and to predict a unimodal probability distribution over bounding boxes.

The KITTI evaluation server computes the AP at a 0.7 IoU across three difficulty levels: easy, moderate, and hard.
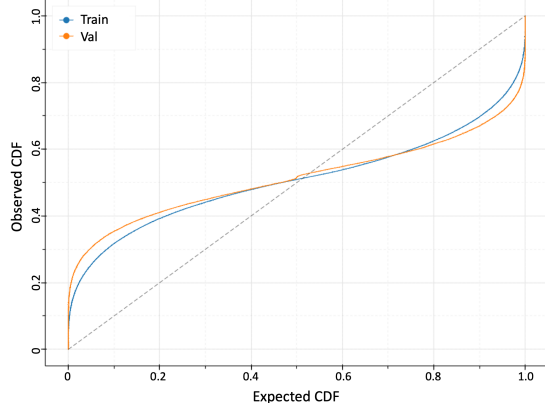
(a) Calibration on KITTI	(b) Calibration on ATG4D

Figure 6: Plots showing the calibration of the predicted distribution over bounding boxes on the train and validation sets. A perfectly calibrated distribution corresponds to a line with unit slope (dashed line in the plots). We observe that the model is unable to learn the probability distribution on KITTI, whereas it is capable of learning the distribution on the larger ATG4D.

As shown in Table 4, our approach performs worse than current state-of-the-art bird's eye view detectors on this small dataset. The following section explores this discrepancy in performance between the results on the large dataset shown in Table 1 and the results on the small KITTI dataset.

4.5. Analysis of the Predicted Distribution

On the small dataset, our approach under-performs compared to state-of-the-art methods. While on a significantly larger dataset, our method out-performs the previous work. The ablation study in Section 4.2 emphasizes the importance of estimating a probability distribution to our approach. If the network is unable to accurately learn this distribution, our method will perform sub-optimally.

To evaluate the quality of the predicted distributions of a learned model, we compute the probability of each ground truth label given our predicted probability distribution, and we plot the expected CDF versus the observed CDF as shown in Figure 6. We perform this evaluation on both the KITTI and the ATG4D datasets. On both datasets, the model is trained to predict a unimodal probability distribution. With the small scale KITTI dataset, our model is unable to correctly estimate the probability distribution. However, the model is capable of precisely learning the distribution on the large scale ATG4D dataset. We hypothesize that learning the distribution requires the network to see many more examples than are available in the KITTI training set, and this helps explain the difference in model performance across these two datasets.

5. Discussion

In recent years, the research community has generally favored detectors that operate in the bird's eye view or di-

rectly on the 3D point cloud. There are many advantages to these approaches. Operating directly in the output space allows a detector to have a consistent prior over the object's shape, which can make the learning problem easier. This is especially important when training on smaller datasets.

However, we think it is a mistake to overlook the range view representation. Deep learning approaches have consistently shown success across many domains when applied to the raw input data without hand-engineered feature extraction, projections, or other manipulations given there is enough training data. The range view is the native representation of the LiDAR data; as a result, it is naturally compact. This compact representation leads to significant gains in efficiency. Moreover, it inherently conveys important contextual information in terms of how the data was captured, and this context can be lost when the range data is projected into a point cloud. At the same time, the range view poses significant challenges to learning due to the varying scale and shape of objects as well as the need to handle occlusions. With a large training set, we have shown that it is possible to overcome these challenges and produce competitive results while being more computationally efficient.

On a smaller dataset, our approach does not achieve the same performance as the state-of-the-art bird's eye view detectors. Key elements of our approach include operating in the native view of the sensor and predicting a probability distribution over bounding boxes. Both of these make the learning problem more difficult and require more training data to perform well.

Finally, although we have focused on one specific representation in this paper, we believe that other detection approaches would also benefit from predicting probability distributions over bounding boxes.

References

[1] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando García, and Arturo De La Escalera. Birdnet: A 3D object detection framework from Li-DAR information. In *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[2] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-NMS – Improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[3] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3D object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3D object proposals for accurate object class detection. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3D vehicle detection. In *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[7] Di Feng, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer. Leveraging heteroscedastic aleatoric uncertainties for robust real-time LiDAR 3D object detection. *arXiv preprint arXiv:1809.05590*, 2018.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[9] Alejandro González, Gabriel Villalonga, Jiaolong Xu, David Vázquez, Jaume Amores, and Antonio M López. Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[10] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2012.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[13] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3D proposal generation and object detection from view aggregation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[16] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3D lidar using fully convolutional network. In *Proceedings of Robotics: Science and Systems (RSS)*, 2016.

[17] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3D object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[18] 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 (ICCV)*, 2017.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[20] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] Cristiano Premebida, João Carreira, Jorge Batista, and Urbano Nunes. Pedestrian detection combining RGB and dense LIDAR data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[22] 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 (CVPR)*, 2018.

[23] 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 (CVPR)*, 2017.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[25] Bin Xu and Zhenzhong Chen. Multi-level fusion based 3D object detection from monocular images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[26] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3D bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] Bin Yang, Ming Liang, and Raquel Urtasun. HDNET: Exploiting HD maps for 3d object detection. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2018.

[28] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3D object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] 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 (CVPR)*, 2018.