# Comparing Fixed-weight and Bayesian Neural networks for 3D Object Detection in the context of Autonomous Driving

## Abstract

*For the safe operation of autonomous agents in the world around, accurate perception and understanding are a must. Object detection is one of the problems in the perception stack to localize and classify the object and this task is being accomplished using Deep Neural Networks with satisfactory results. But, the neural networks are deterministic. Thereby, providing overconfident results which lead to some catastrophic failures. On the other hand, Bayesian Neural Network can provide an extra measure of uncertainty along with the object proposals. In this work, we choose to compare the performances of the Fixed-weight Neural Networks and Bayesian Neural Networks on the task of localizing and classifying the objects in a 3D world. In the results, we show that the fixed-weight neural networks were over-confident in detecting some objects. While Bayesian networks do provide a measure of uncertainty which represents that the model is not certain.*

## 1. Introduction

Perceiving the outside environment consists of tasks like lane detection, vehicle detection, and Vulnerable Road Users (VRU)[1] detection. As the vehicles are operating in the 3D world, for better autonomous operation, the perception part assists in knowing the 3D position of the agents[2] around [**?**]. This task has to be carried out with high accuracy and should be robust to adverse weather and lighting conditions.

The task of object detection is carried out with the usage of Deep Neural Networks(DNNs) [**?**, **?**] that use Cameras [**?**, **?**, **?**, **?**], LiDAR [**?**, **?**, **?**, **?**, **?**] or a fusion of both [**?**, **?**, **?**, **?**]. This research is empowered by the datasets from KITTI [**?**], Waymo[**?**], nuTonomy[**?**].

But, there are failures while using DNNs in safety-related tasks. For example, the image-classification model by Google had wrongly classified photos of some humans as gorillas [**?**]. Another such failure was reported when a self-driving car from Uber hit a jay-walking pedestrian and resulted in death [**?**]. Both of these examples show that DNNs are prone to failures. Hence, their usage in safety-critical operations is hindered.

Deep Neural Networks (DNNs), since their inception, have out-performed the conventional statistical approaches in both accuracy and generalizability. But they cannot quantify uncertainty in the outputs. The classification models provide predictive probabilities which are softmax outputs of a model. But the softmax probabilities are wrongfully interpreted as an uncertainty metric. which leads to false positives with very high scores [**?**, **?**]. This sort of behavior is unsolicited in applications like autonomous driving. Hence, These methods are not compatible with safety-critical applications.

Bayesian Neural Networks (BNNs) [**?**, **?**, **?**] are a class of neural networks in which the weights are initialized as a distribution $p(w)$ and training includes obtaining a posterior distribution over weights represented by $p(w|D)$, where $D$ is the dataset on which the model is trained. $p(w|D)$ captures the model uncertainty. Additionally, BNNs also permit the knowledge of uncertainty source [**?**]. The predictive uncertainty is classified into epistemic uncertainty and aleatoric uncertainty [**?**]. Aleatoric uncertainty is further classified into Homoscedastic uncertainty, which stays constant for different inputs and Heteroscedastic uncertainty varies with input, and some being more affected because of the noise from the environment.

### 1.1. Contributions

In this work, we develop Flipout-based Bayesian neural network [**?**] methods to capture epistemic uncertainty in a Frustum-PointNet based 3D object detector. Our contributions are three-fold:

- Can a BNNs version of the 3D object detection be modeled and trained?
- Does the Bayesian model perform at the same level as a Fixed-Weight model?

[1]Vulnerable Road Users (VRU) are defined in the European Union Intelligent Transportation Systems Directive as "non-motorized road users, such as pedestrians and cyclists as well as motorcyclists and persons with disabilities or reduced mobility and orientation".

[2]Agents in this work refers to a car, pedestrians, and cyclist present in the operating environment

- Can the outcomes of the Bayesian model be explained?

Along with answering the above questions we also provide a literature review of the 3D object detection methods and uncertainty quantification techniques using BNNs.

## 2. Related Works

In this section, we summarize various 3D object detection methods and different bayesian neural network architectures for extracting uncertainty using the chosen 3D object detection network.

### 2.1. 3D object detection in Autonomous Driving

The main aim of the 3D object detection model is to take RGB image along with depth information from sensors like Light Detection and Ranging (LiDAR), cameras, Radio Detection and Ranging (RADAR), and output a 3D bounding box along with class labels of the agents in the perceived environment. It is also referred to as amodal 3D object detection [?], which means the model should fit a 3D bounding box for the whole object even though only a part of it is visible.

The best-performing methods for 3D object detection are majorly fusion-based because of the dense input similar to human vision provided by the camera and the depth information provided by the LiDAR point cloud and combining both of them helps to understand the depth of the scene as well as its semantics.

In the work reported by *Du et al.* [?], a pre-trained 2D detection network proposes a candidate for a bounding box this information is fused with the 3D features from the point cloud, and the 3D parameters are regressed with it. MV3D [?] by *Chen et al.*, in which three inputs RGB image, point cloud projected onto the front view, and BEV point cloud are consumed. The front view projection provides information about height, intensity, and distance and the BEV images hold density, intensity, and height information. Then an RPN generates 3D object proposals from BEV only and these proposals are projected to multiple views. AVOD is proposed by *Ku et al.* [?], a deep continuous fusion paradigm is utilized where a 2D BEV projection map is fed to a CNN and another CNN extracts feature maps from the monocular image. These features are fed to a parallel network, which consists of an RPN and a detection network. Frustum-PointNet proposed by *Qi et al.* [?], where a 2D object detector is used to generate a detection in 2D and this 2D detection is used to reduce the search space to process only the required part of the point cloud where the object presence is given. The frustum generated is fed to an encoder-decoder-based instance segmentation network which assigns the labels to the points in the point cloud based on the class label. This helps in finding the objects when there is a presence of partial occlusion. The segmented point cloud is passed to a PointNet [?] based amodal bounding box estimation which regresses the 3D bounding boxes.

### 2.2. Bayesian Neural Networks

In conventional DNNs, weights are modeled as point estimates which is unjustified. These point estimates resulted in the network being deterministic for provided inputs. Whereas, Bayesian Neural Networks (BNNs) models the weights as a distribution of weights defined over the given dataset $D$. This helps in extracting uncertainty by sampling weights from the posterior distribution of weights.

Let us consider a dataset $D = \left\{\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right)\right\}$, and $w$ be the weights of the neural network that are to be learnt from the dataset $D$ provided. The posterior distribution $p(w \mid D)$ is obtained by applying Bayes rule as in Equation 1

$$p(w \mid D) = \frac{p(D \mid w)p(w)}{p(D)} \tag{1}$$

Calculating the denominator $p(D)$ is not manageable because of the integration over all the weights. The DNNs generally have millions of weights and integrating all over them is tedious and is computationally intensive. Hence, we estimate the $p(w \mid D)$ distribution.

*Graves et al,* [?] proposed that the posterior distribution $p(w \mid D)$ can be approximated to be a parameterized distribution $q(w)$ and optimize those distributions over the weight parameters to obtain the best approximation.

The approximation is judged using Kullback-Leibler (KL) divergence [?] as shown in Equation 2.

$$\begin{aligned}\mathrm{KL}(q(w)\|p(w \mid D)) = \mathrm{KL}(q(w)\|p(w)) + \log p(D)\\ -\mathbb{E}_{q(w)}[\log p(D \mid w)]\end{aligned} \tag{2}$$

Hence, VI poses the problem of finding the $p(w \mid D)$ into a maximizing ELBO problem. Once the optimization problem is defined we can use Bayes by backprop by *Shridhar et al.* [?], to extract the best approximation of posterior which maximizes the ELBO. Bayes by Back-propagation was first introduced by *Blundell et al.* [?], which allowed the usage of current back-propagation algorithm work by replacing the weights with a parameterized distribution. Hence, the optimal parameters $\theta^{opt}$ which consists of respective mean and variance $(\mu, \sigma)$ are defined as

$$\begin{aligned}\theta^{opt} = \operatorname{argmin}_\theta \mathbf{KL}[q(\mathbf{w} \mid \theta)\|P(\mathbf{w})] - \\ \mathbb{E}_{q(\mathbf{w}\mid\theta)}[\log P(\mathfrak{D} \mid \mathbf{w})]\end{aligned} \tag{3}$$

Substituting this in Equation 3 we get,

$$\theta^* = \operatorname{argmin}_\theta \mathbb{E}_{q(\mathbf{w}\mid\theta)} f(\mathbf{w}, \theta) \tag{4}$$

But, the above objective function cannot be used with a standard back-propagation algorithm. This is due to the inability to achieve differentiation of the objective function. Hence, *Kingma et al* [**?**] proposed a reparameterization trick in which the gradients are calculated through general back-propagation and also reducing the variance. In case of CNNs, *Shridhar et al.* proposed to sample layer activations instead of weights, it results in a estimated posterior distribution $q_\theta\left(\theta_{\mathrm{i}jhw} \mid Data\right)$.The resultant equation for sampling the activation is written as

$$b_j = A_{\mathrm{i}} * \mu_{\mathrm{i}} + \varepsilon_j \cdot \sqrt{A_{\mathrm{i}}^2 * \left(\alpha_{\mathrm{i}} \cdot \mu_{\mathrm{i}}^2\right)} \tag{5}$$

$b$ is the layer activation, $\varepsilon_j \epsilon N(0,1)$, $A$ is the receptive field area.

*Flipout based Weight-sampling methods:* To perform inference for a new input $x^*$ given a VI approximation $q(w)$ there is a need to sample the weights from the approximation for every inference run performed to extract $p\left(y^* \mid x^*, w\right)$.using Flipout [**?**] introduced in *Wen et al.*. This method aims to uncorrelated the gradients between every sample. This work achieved by making some assumptions that are

1. The sampling of different weights is independent of each other.
2. The posterior distribution is symmetric around 0.

Flipout suggests using a posterior distribution ($q_\theta$) which is jointly used by every weight sample on the mini-batch and an element-wise multiplication is performed by another rank-one sign matrix for every sample. A random sign matrix $E$ which is independent of posterior distribution $q_\theta$. The weight sampling is done by multiplying $E$ and the posterior distribution

$$W = q_\theta, E$$

.

With the availability of the approximation $q(w)$ of $p(w \mid D)$ obtained using the VI method we can infer an output $y^*$ for any given input $x^*$ by integrating over the weights available as shown in Equation 6.

$$p\left(y^* \mid x^*, D\right) \approx \frac{1}{K} \sum_{k=1}^{K} p\left(y^* \mid x^*, w^k\right) \tag{6}$$

## 3. Epistemic Uncertainty Quantification

In this section, we will summarize the method to quantify the uncertainty in 3D object detection.

### 3.1. Quantifying Epistemic Uncertainty in Classification

Uncertainty in classification can be quantified using the Shannon entropy of the final softmax scores in Equation 8. As used by *Feng et al.* [**?**]. For an input of $x^*$, the classification probability of class for N with the availability of the probability score is calculated using Equation 7. This probability is used to calculate Shannon entropy (SE) using Equation 8

$$\begin{aligned} p\left(c \mid \mathbf{x}^*\right) &= \mathbb{E}_{p(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})} p\left(c \mid \mathbf{x}^*, \mathbf{W}\right) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} s_{\mathbf{x}^*}^i \end{aligned} \tag{7}$$

$$\begin{aligned} &SE\left(\mathbf{y}^* \mid \mathbf{x}^*\right) = \mathbb{H}\left(\mathbf{y}^* \mid \mathbf{x}^*\right) \\ &= -p\left(c \mid \mathbf{x}^*\right) \log p\left(c \mid \mathbf{x}^*\right) - p\left(\neg c \mid \mathbf{x}^*\right) \log p\left(\neg c \mid \mathbf{x}^*\right) \\ &\approx -\tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}}^i \cdot \log \tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}^*}^i - \\ &\left(1 - \tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}^*}^i\right) \log \left(1 - \tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}^*}^i\right) \end{aligned} \tag{8}$$

If the SE value is zero, then the network is most certain of the output and when the SE value is higher the network is not certain of the classification result.

### 3.2. Quantifying Epistemic Uncertainty in Regression

To find the spatial uncertainty in the bounding box parameters *Feng et al.* [**?**] proposed to calculate Total Variance (TV), which represents how much each measurement varies from their mean value.

1. Transform the bounding box detections into world reference frame and the mean value of the detections regressed during N forward passes is calculated.
2. Covariance matrix of the detections is calculated using the following the equation $C\left(\mathbf{x}^*\right) = \frac{1}{N} \sum_{i=1}^{N} \hat{\mathbf{v}}_{\mathbf{x}^*}^i \hat{\mathbf{v}}_{\mathbf{x}^*}^{i^T} - \mathbf{I}_{\mathbf{x}^*} \mathbf{I}_{\mathbf{x}^*}^T$.
3. Total variance is calculating by taking the trace of the covariance matric $C(x^*)$.

The Total variance spans from $[0, +\infty)$, the higher the value represents the higher the epistemic uncertainty in regressed bounding box parameters.

## 4. Experimental Results

In this section, we explain various experiments, their results, and comparing the performance of different networks.

### 4.1. Training Frustum-PointNet

The main aim of this experiment is to train a Frustum-PointNet using Lyft dataset. The model is trained using a total of 230,944 samples divided into 7,217 training samples of batch size 32. The training is done for 178 (i,e. Three days) epochs.

Table 1. 3D Average Precision values for 3D object detection on Lyft dataset [?]

| Difficulty | Car (%) | Cyclist (%) | Pedestrian (%) |
|---|---|---|---|
| **Easy** | 80.19 | 69.79 | 60.59 |
| **Moderate** | 64.74 | 42.35 | 34.93 |
| **Hard** | 58.28 | 37.28 | 34.93 |

## 4.2. Bayesian Neural network modelling and evaluation

We converted the Frustum-PointNet model into a Bayesian model using TensorFlow probability library. But, we observed that the model is under-fitting. We converted the Frustum-PointNet model into partial-Bayesian model, by fixing the weights of instance segmentation model and converting the spatial-transformer network and bounding box network into Bayesian model which fitted during training.

Table 2. 3D Average Precision values for 3D object detection on Lyft dataset [?]

| Difficulty | Car (%) | Cyclist (%) | Pedestrian (%) |
|---|---|---|---|
| **Easy** | 80.32 | 69.87 | 60.43 |
| **Moderate** | 68.74 | 48.35 | 38.93 |
| **Hard** | 59.8 | 36.28 | 39.93 |

## 4.3. Epistemic Uncertainty Quantification and Analysis

the epistemic uncertainty in the 3D object detection task calculated from the multiple Monte Carlo runs. During these inference runs, weights are sampled from the distribution $p(w)$. The resultant models hold different sampled weights and does form an ensemble like architecture and helps in extracting epistemic uncertainty. The epistemic uncertainty should be quantified using Shannon entropy and Total variance. We used the bounding box dimensions, bounding box center and rotation angle to calculate the epistemic uncertainty.

The shannon entropies calculated using the softmax probabilities at each stage and are plotted as follows

Table 3. Epistemic uncertainty statistics from the multiple Monte-Carlo runs which shows that the epistemic uncertainty is higher for the pedestrian and the cyclist

| Agent | Car | Cyclist | Pedestrian |
|---|---|---|---|
| **No of samples** | 16314 | 3824 | 5254 |
| **mean variance** | 1.34 | 4.48 | 2.88 |
| **mean shannon entropy** | 0.43 | 0.88 | 0.93 |

F$_{Pointnet_{R}esults/Entropy_{C}ar.pdf}$ latex/images/Part-Ba

Figure 1. Entropy for Car detection

F$_{Pointnet_{R}esults/Entropy_{C}yclist.pdf}$ latex/images/Part-B

Figure 2. Entropy for Cyclist detection

F$_{Pointnet_{R}esults/Entropy_{P}edestrian.pdf}$ latex/images/Part-

Figure 3. Entropy for Pedestrian detection

## 4.4. Visualizations

### 4.4.1 Uncertainty due to agents blocking one other

The epistemic uncertainty of the car marked is high because of the car marked in green is blocking it in Image 12. Number of points available are very low in the bounding box, but the detection is made because of the bin size from the class in 2D object detection.

latex/.

Figure 4. Predictions in image plane.

### 4.4.2 Epistemic uncertainty due to cluttered environments

The epistemic uncertainty in a cluttered environment is high in case of every object present in the clutter the entropy in some cases is as high as its maximum value i,e 1.0. The total variance for every object as well is very high when compared to a non-cluttered scene. Because of the overlaps in the detection, the point cloud of one object is affecting

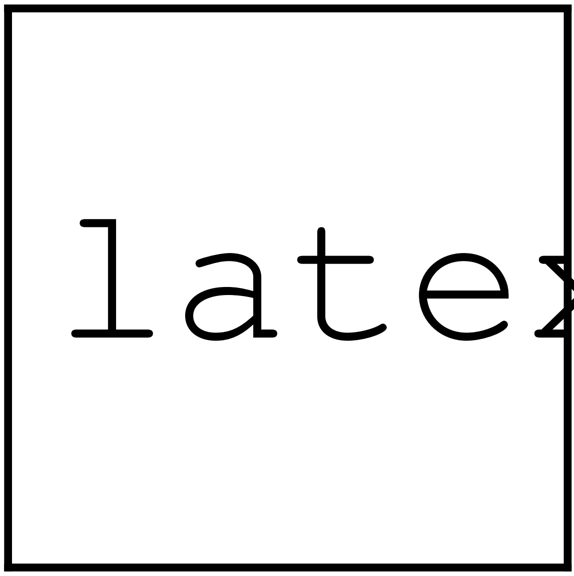
Figure 5. Predictions in LiDAR plane.

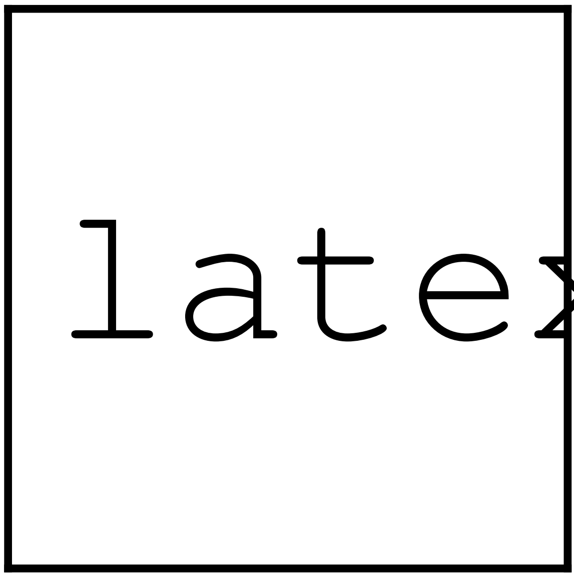
Figure 6. Predictions in BEV projection of LiDAR point cloud.

the other underlying objects as well which results in high entropy outputs

#### 4.4.3 Epistemic uncertainty due to distance to objects

The epistemic uncertainty of the car marked in red is at a distance from the ego-vehicle which might result in

- Reduced number of points in the extracted point cloud.
- Increased in the angle between centroid of the point cloud and the heading angle of the frustum point cloud, which might result in a overlapped bin size

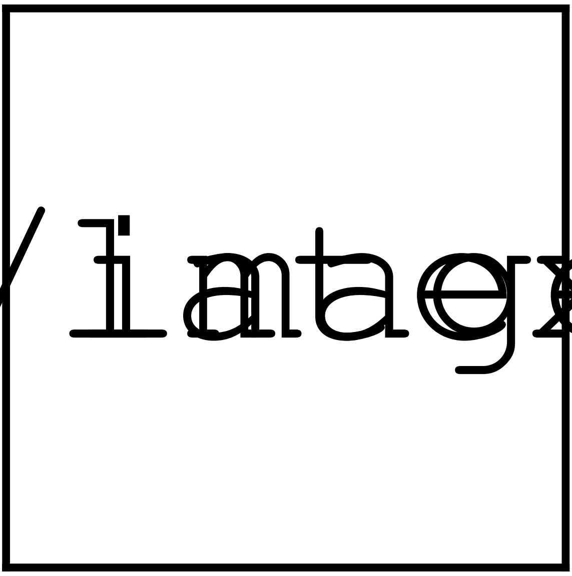
Figure 7. Predictions in image plane.

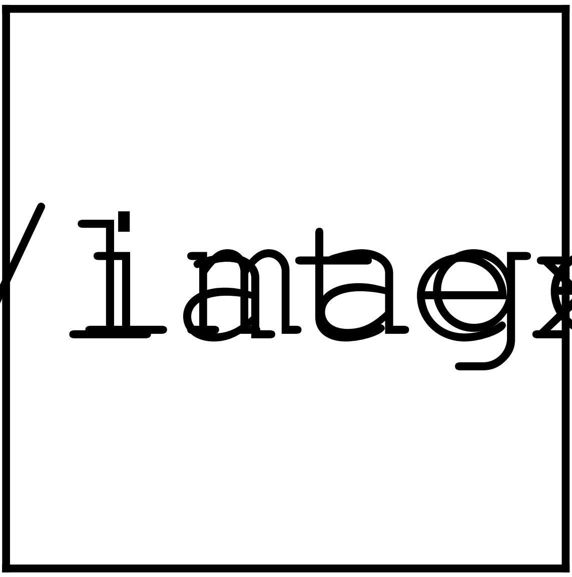
Figure 8. Predictions in LiDAR plane.

## 5. Conclusion and Future works

In this research and development project, Frustum-PointNet [?] is modeled to solve the 3-Dimensional (3D) object detection problem in the context of Autonomous driving. The network is trained and tested on Lyft dataset [?]. We observed that the network performed on the same level as the network trained using KITTI dataset [?]. The comparison is done based on 3D Average-Precision values calculated using fixed limits set on Intersection-over-Union values.

We also modeled a Bayesian Neural Network based on Frustum-PointNet [?] to extract the uncertainty due to the

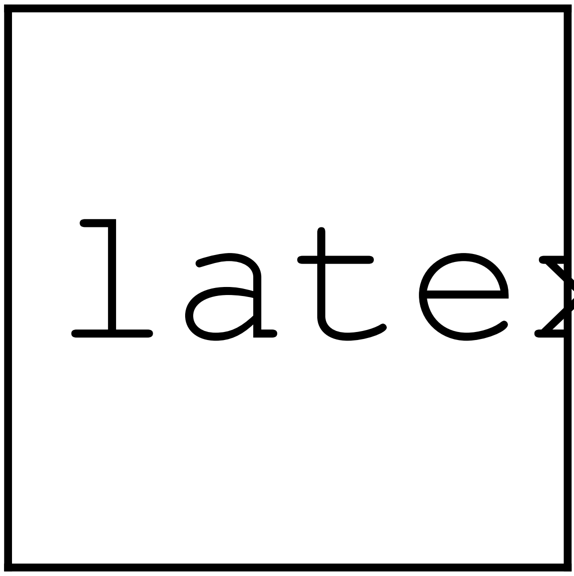

Figure 9. Predictions in BEV projection of LiDAR point cloud.

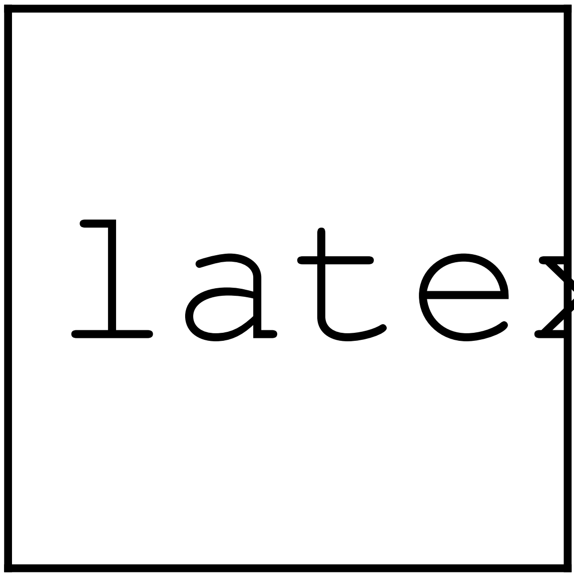

Figure 10. Predictions in image plane.

weights in the Frustum-PointNet [?] model. We also concluded that the Bayesian Neural Network performed better than the Fixed-weight model. But, we observed that the difference in performance is not as expected. We believe that with the better tuning of the prior distribution placed on the weights, we can achieve better results.

Further analysis on the extracted uncertainty also showed that the model is over-confident over some samples. We also visualized some samples which represented the model's over-confidence and confirm the need for the usage of uncertainty as a measure of confidence.

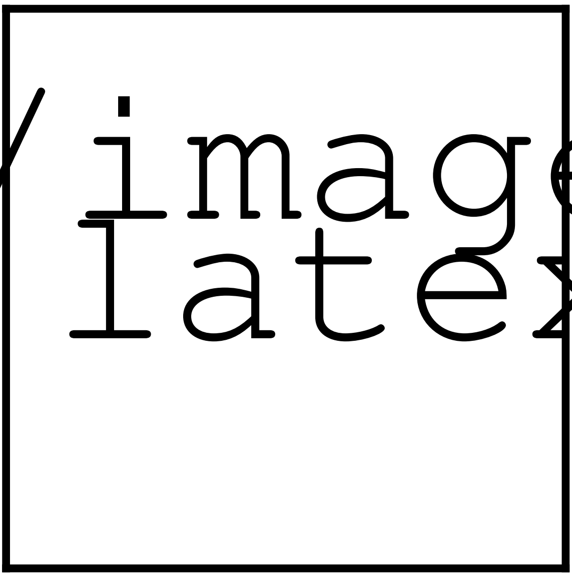

Figure 11. Predictions in LiDAR plane.

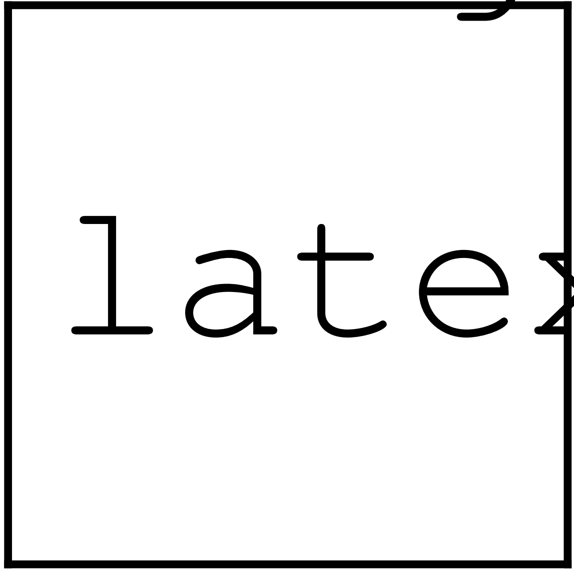

Figure 12. Predictions in BEV projection of LiDAR point cloud.