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

Anonymous CVPR submission

Paper ID ****

Abstract

Accurate perception and understanding of a scene is a pre-requisite for the safe operation of autonomous agents. Current state-of-the-art methods use Deep Neural Networks (DNN) in order to localize and classify objects. However, the parameters of neural networks are often point estimates which often provide overconfident results. This may lead to catastrophic failures, especially in an application such as autonomous driving. In contrast, Bayesian Neural Networks (BNNs) can provide an additional measure of uncertainty together with the object proposals. Thus, in this work we compare the performance of fixed-weight neural networks versus BNNs in the task of localization and classification of objects in 3D space. The results show that fixed-weight neural networks are indeed over-confident in 3D detection task, while BNNs provide an uncertainty estimation, while still retaining the same accuracy.

1. Introduction

In order to properly perceive the environment, autonomous cars must be able to reliably solve tasks such as lane detection, vehicle detection, as well as Vulnerable Road Users (VRU)[1] detection. Thus, autonomous cars require the vehicle's perception module to predict the 3D position of the agents[2] in the proximity [11]. This task has to be carried out with high accuracy and should be robust against adverse weather, occlusions and extreme lighting conditions.

The task of object detection is carried out with Deep Neural Networks (DNNs) [22, 1] that use RGB cameras [6, 20, 4, 35], LiDAR [9, 34, 24, 27, 33] or a fusion of both [8, 21, 17, 36]. Furthermore, advances in this research have been enabled by datasets such as KITTI [11], Waymo [29], and nuTonomy [3].

However, failures do occur using DNNs in safety-related tasks. The image-classification model by Google, for example, incorrectly classified photos of humans as gorillas [19]. Another such failure occured when a self-driving car hit a jay-walking pedestrian resulting in the pedestrians death [25].

DNNs out-perform the conventional statistical approaches in both accuracy and generalizability. However, they alone cannot quantify the uncertainty in the outputs. The classification models provide predictive probabilities which are softmax outputs of a model. However, the softmax probabilities are incorrectly interpreted as an uncertainty metric. which leads to false positives with very high scores [2, 18]. This sort of behaviour is unsolicited in applications such as autonomous driving, rendering these methods incompatible with safety-critical applications.

BNNs [31, 26, 30] 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 sources [13]. The predictive uncertainty is classified into epistemic uncertainty and aleatoric uncertainty [14]. Aleatoric uncertainty is further classified into Homoscedastic uncertainty, which remains constant for different inputs and Heteroscedastic uncertainty which varies with input.

1.1. Contributions

In this work, a Flipout-based Bayesian Neural Network [32] architecture is developed to capture the uncertainty of a Frustum-PointNet for 3D object detection. The main achievements in this paper are:

- BNNs and Frustum-PointNets have been successfully combined for 3D object detection.
- Marginally better results were obtained using the

[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

Bayesian model versus the fixed-weight architecture.

- Both a quantitative and a qualitative depiction of uncertainty are provided.

In addition to the above contributions, a literature review of the 3D object detection methods and the uncertainty quantification techniques used in BNNs is given.

2. Related Work

In this section, various 3D object detection methods, as well as different BNN architectures for extracting uncertainty using the chosen 3D object detection network are summarized.

2.1. 3D object detection in Autonomous Driving

The main aim of the 3D object detection model is to process 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 [1], 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, as reported by KITTI Benchmark tests [11]. This results are possible by the dense input provided by the camera and the depth information provided by the LiDAR point cloud. The combination of these orthogonal data sources provides features to predict the depth of the objects as well as its semantics.

In the work reported by *Du et al.* [8], 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, in order to regress the 3D parameters. *Chen et al.* proposed MV3D [7], 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.* [17], 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.* [21], 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 [5] based amodal bounding box estimation which regresses the 3D bounding boxes. The architecture of Frustum-PointNet [21] is portrayed in Figure 1.

2.2. Bayesian Neural Networks

In conventional DNNs, weights are modeled as point estimates. 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 possible because of the integration over all the weights. The DNNs generally have millions of weights and integrating all over them is computationally expensive. Hence, we approximate the $p(w \mid D)$ distribution.

Graves et al, [12] 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 [28] 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.* [23], to extract the best approximation of posterior which maximizes the ELBO. Bayes by Back-propagation was first introduced by *Blundell et al.* [2], which allowed the usage of current back-propagation algorithm work by replacing the weights with a parameterized distribution. Hence, the optimal parameters θ^{opt} which consists of respective mean and variance (μ, σ) are defined as

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

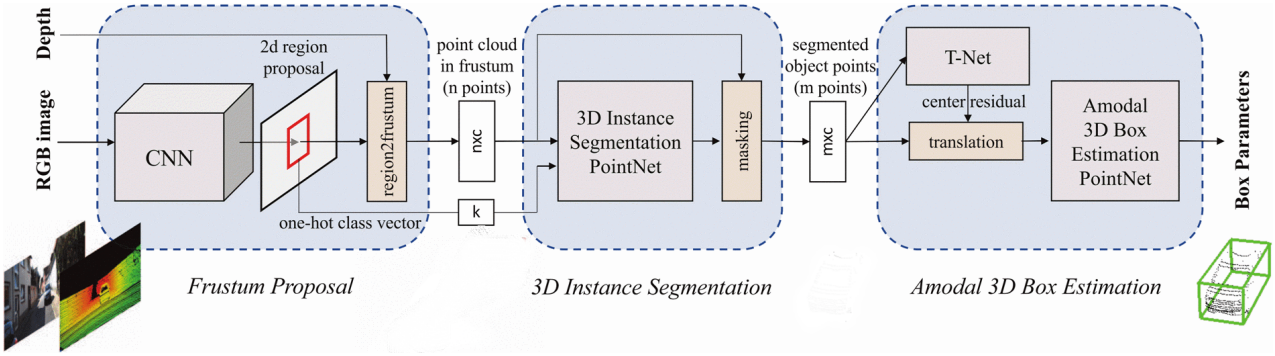

Figure 1. Frustum PointNet for 3D Object Detection in Autonomous driving.

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 stochastic node inside the objective function. Hence, *Kingma et al* [16] proposed a reparameterization trick by representing the stochastic node as a point value parameterized by a distribution from which a scale value for the weights can be sampled from. This resulted in gradient calculation through general back-propagation.

In case of CNNs, *Shridhar et al.* proposed to sample layer activations instead of weights, this results in a estimated posterior distribution $q_\theta\left(\theta_{ijhw} \mid Data\right)$.The resultant equation for sampling the activation is written as

$$b_j = A_{\mathrm{i}} * \mu_{\mathrm{i}} + \varepsilon_j \cdot \sqrt{A_{\mathrm{i}}^2 * \left(\alpha_{\mathrm{i}} \cdot \mu_{\mathrm{i}}^2\right)} \tag{4}$$

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

Flipout based Weight-sampling methods: Though the reparameterization method is able to achieve an approximation $q(w)$ by backpropogation. But the resultant weight updates in batch are correlated, resulting in updates with high variance. Flipout is introduced by *Wen et al.* [32], it aims to decorrelate the gradients between every sample in a mini-batch. Decorrelation is achieved by making the following assumptions:

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_θ) 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_θ. 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 5.

$$p\left(y^* \mid x^*, D\right) \approx \frac{1}{K} \sum_{k=1}^{K} p\left(y^* \mid x^*, w^k\right) \tag{5}$$

3. Uncertainty Quantification

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

3.1. Quantifying Epistemic Uncertainty in Agent Classification

Uncertainty in classification can be quantified using the Shannon entropy of the final softmax scores in equation 7. As used by *Feng et al.* [10], for an input of x^*, the classification probability of class for N monte-carlo inference runs is calculated using equation 6. This probability is used to calculate Shannon entropy (SE).

$$\begin{aligned} p\left(c \mid \mathbf{x}^*\right) &= \mathbb{E}_{p(\mathbf{W}|\mathbf{X},\mathbf{Y})} p\left(c \mid \mathbf{x}^*, \mathbf{W}\right) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} s_{\mathbf{x}^*}^i \end{aligned} \tag{6}$$

$$\begin{aligned} &SE\left(\mathbf{y}^* \mid \mathbf{x}^*\right) = \mathbb{H}\left(\mathbf{y}^* \mid \mathbf{x}^*\right) \\ &= -p\left(c \mid \mathbf{x}^*\right) \log p\left(c \mid \mathbf{x}^*\right) - p\left(\neg c \mid \mathbf{x}^*\right) \log p\left(\neg c \mid \mathbf{x}^*\right) \\ &\approx -\tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}}^i \cdot \log \tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}^*}^i - \\ &\left(1 - \tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}^*}^i\right) \log \left(1 - \tfrac{1}{N} \textstyle\sum_{i=1}^{N} s_{\mathbf{x}^*}^i\right) \end{aligned} \tag{7}$$

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 uncertain of the classification result.

3.2. Quantifying Total-Uncertainty in Regression

To find the spatial uncertainty in the bounding box parameters *Feng et al.* [10] proposed to calculate Total Vari-

ance (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 $C(\mathbf{x}^*) = \frac{1}{N}\sum_{i=1}^{N} \hat{\mathbf{v}}^i_{\mathbf{x}^*} \hat{\mathbf{v}}^{i^T}_{\mathbf{x}^*} - \mathbf{I}_{\mathbf{x}^*}\mathbf{I}^T_{\mathbf{x}^*}$.

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

4.1. Training Frustum-PointNet

The Frustum-PointNet model is trained for 178 epochs on the frustums extracted. The loss values and 3D IoU values obtained during validation step are visualized in the Figure 2 and Figure 3. The quantitative evaluation results of the Frustum PointNet is mentioned in Table 1

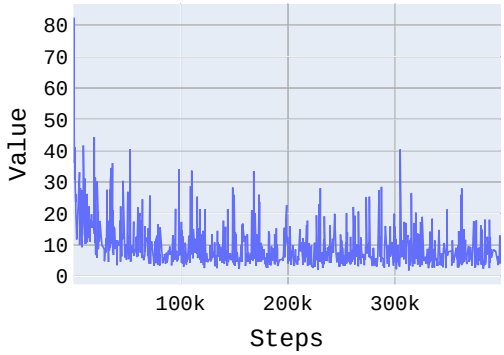

Figure 2. The loss values collected during training the Frustum PointNet

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

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

4.2. Bayesian Neural network modelling and evaluation

We converted the Frustum-PointNet model into a Bayesian architecture using TensorFlow probability library

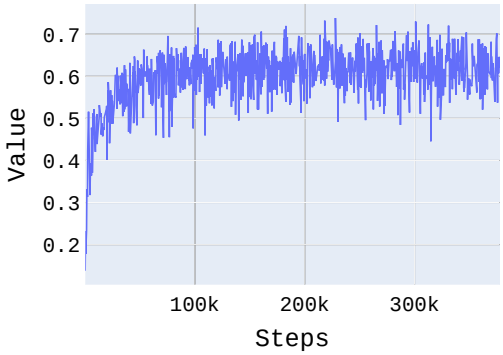

Figure 3. The 3D IoU values calculated during the validation step of the training process

[30]. But, we observed that the model is severely under-fitting. From the Figure 4 which portrays the loss value during the Bayesian Frustum PointNet training it is inferred that the model is under-fitting.

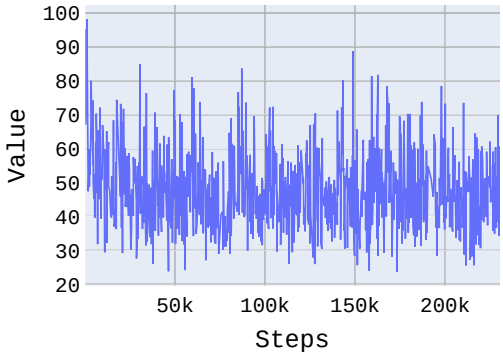

Figure 4. The loss values collected during training the Bayesian Frustum PointNet

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 architecture which fitted well during training.

Comparing Figures 3, 6 and Figures 2, 5 we can also observe the regularization effect of KL-Divergence between prior weight values and the posterior distribution leading to reduction in the amplitude of oscillations observed in both loss and IoU values.

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

Table 2. 3D Average Precision values for 3D object detection on Lyft dataset [15]

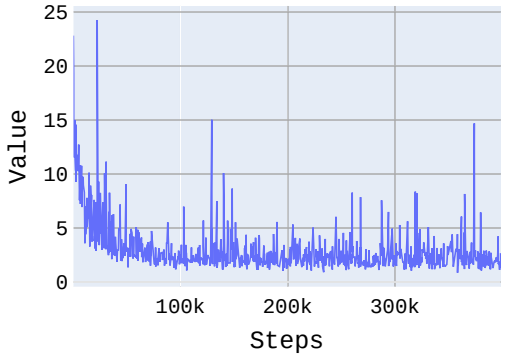

Figure 5. The loss values collected during training the partial Bayesian Frustum PointNet

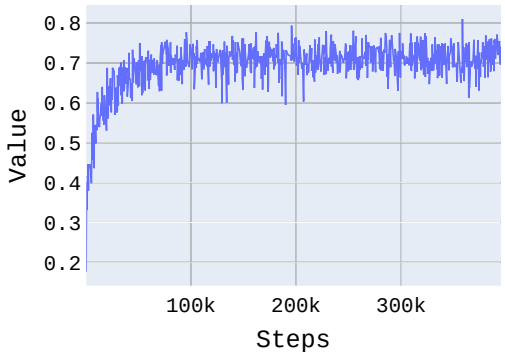

Figure 6. The 3D IoU values collected during validation step of training the partial Bayesian Frustum PointNet

4.3. Epistemic Uncertainty Quantification and Analysis

The epistemic uncertainty in the 3D object detection task is calculated from the multiple Monte Carlo runs. During these inference runs, weights are sampled from the approximated distribution $q(w)$. The resultant models hold different sampled weights and form an ensemble like architecture and helps in extracting epistemic uncertainty. The epistemic uncertainty is 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

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 Figure 18. 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.

Agent	Car	Cyclist	Pedestrian
No of samples	16314	3824	5254
Total Uncertainty	1.34	4.48	2.88
Epistemic Uncertainty	0.43	0.88	0.93
Aleatoric Uncertainty	0.91	3.6	1.95

Table 3. Uncertainty statistics from the multiple Monte-Carlo runs which shows that the epistemic uncertainty is higher for the pedestrian and the cyclist. Since, the samples are from the same dataset used for training we considered the data-domain variance to be zero

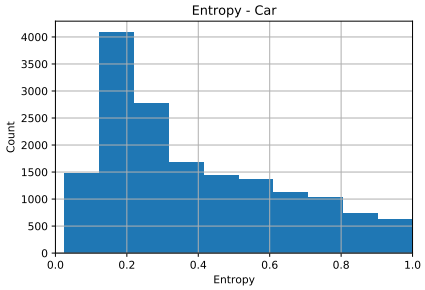

Figure 7. Entropy for Car detection

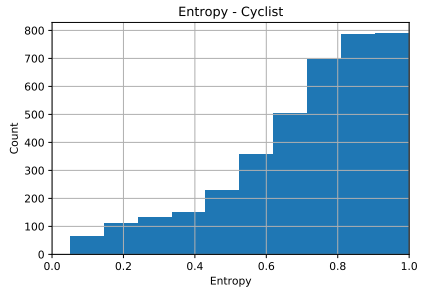

Figure 8. Entropy for Cyclist detection

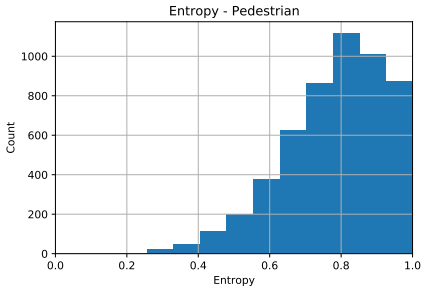

Figure 9. Entropy for Pedestrian detection

4.4.2 Epistemic uncertainty due to cluttered environments

The epistemic uncertainty in a cluttered environment is high for every object present in the clutter. The total variance for

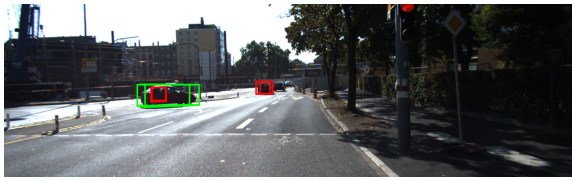

Figure 10. Predictions in the image plane.

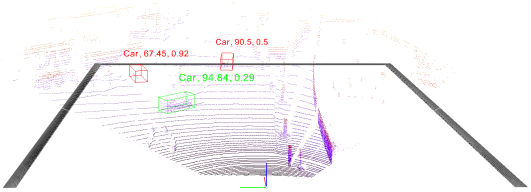

Figure 11. Predictions in the LiDAR plane.

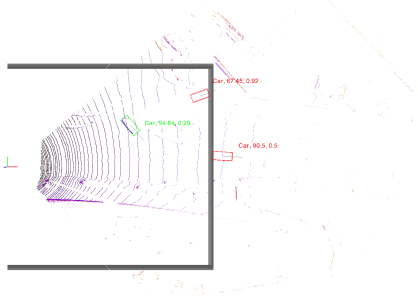

Figure 12. Predictions in the BEV projection of the LiDAR point cloud.

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

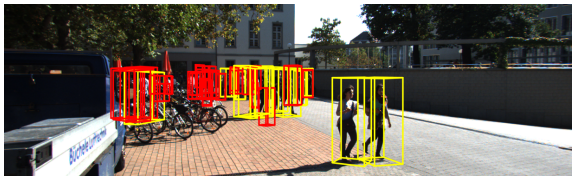

Figure 13. Predictions in the image plane.

Figure 14. Predictions in the LiDAR plane.

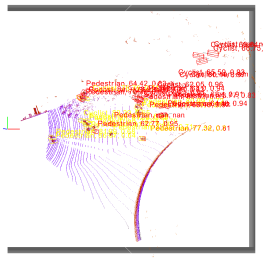

Figure 15. Predictions in the BEV projection of the LiDAR point cloud.

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 16. Predictions in the image plane.

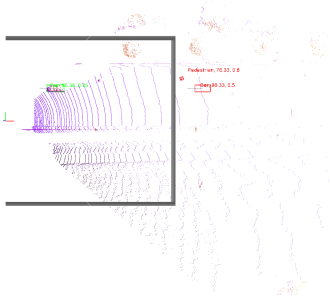

Figure 17. Predictions in the LiDAR plane.

5. Conclusion and Future works

In this paper, A fixed-weight Frustum-PointNet [21] architecture is trained to solve the 3-Dimensional (3D) object detection problem in the context of autonomous driving. The network is trained and tested on the Lyft dataset [15]. It was observed that the network performed on the same level as the network trained using the KITTI dataset [11]. This comparison is done based on 3D Average-Precision values

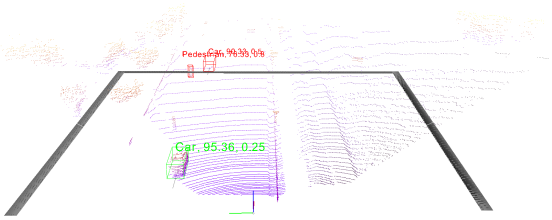

Figure 18. Predictions in the BEV projection of the LiDAR point cloud.

calculated using fixed limits set on Intersection-over-Union values.

Furthermore, a Bayesian Neural Network is modelled based on the Frustum-PointNet [21] to extract the uncertainty due to the weights. It could be concluded that the BNN performed better than the fixed-weight model.

Further analysis on the extracted uncertainty showed that the point-wise model is over-confident. Some samples which represent the model's over-confidence are visualized. These results confirm the need for the usage of uncertainty as a measure of confidence.

References

[1] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, Oct 2019.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1613–1622. JMLR.org, 2015.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[4] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1827–1836, 2017.

[5] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[6] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2147–2156, 2016.

[7] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D Object Detection Network for Autonomous Driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6526–6534, 2017.

[8] Xinxin Du, M. Ang, Sertac Karaman, and D. Rus. A general pipeline for 3d detection of vehicles. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3194–3200, 2018.

[9] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361, 2017.

[10] D. Feng, L. Rosenbaum, and K. 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, Nov 2018.

[11] A Geiger, P Lenz, and R Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

[13] Maximilian Henne, Adrian Schwaiger, Karsten Roscher, and Gereon Weiss. Benchmarking uncertainty estimation methods for deep learning with safety-related metrics. In Huáscar Espinoza, José Hernández-Orallo, Xin Cynthia Chen, Seán S. ÓhÉigeartaigh, Xiaowei Huang, Mauricio Castillo-Effen, Richard Mallah, and John McDermid, editors, *Proceedings of the Workshop on Artificial Intelligence Safety, co-located with 34th AAAI Conference on Artificial Intelligence, SafeAI@AAAI 2020, New York City, NY, USA, February 7, 2020*, volume 2560 of *CEUR Workshop Proceedings*, pages 83–90. CEUR-WS.org, 2020.

[14] Kendall, Alex, Gal, Yarin. 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.

[15] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft Level 5 AV Dataset 2019. online, 2019.

[16] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

[17] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[18] Malinin, Andrey, Gales, Mark. Predictive uncertainty estimation via prior networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7047–7058. Curran Associates, Inc., 2018.

[19] Molly Mulshine. A major flaw in Google's algorithm allegedly tagged two black people's faces with the word 'gorillas'. Online, 2015. (visited on 08/13/2020).

[20] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. 3d bounding box estimation using deep learning and geometry. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5632–5640, July 2017.

[21] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[22] Rao, Qing, Frtunikj, Jelena. Deep learning for self-driving cars: Chances and challenges. In *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, SEFAIS '18, page 35–38, New York, NY, USA, 2018. Association for Computing Machinery.

[23] S. Kumar, L. Felix and L. Marcus. Uncertainty Estimations by Softplus normalization in Bayesian Convolutional Neural Networks with Variational Inference. *arXiv preprint arXiv:1806.05978,*, 2019.

[24] R. Sahba, A. Sahba, M. Jamshidi, and P. Rad. 3d object detection based on lidar data. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0511–0514, Oct 2019.

[25] Sam Levin, Julia C Wong. Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian. Online, 2018. visited on 08/12/2020.

[26] K. Shridhar, F. Laumann, A. L. Maurin, Martin Olsen, and Marcus Liwicki. Bayesian convolutional neural networks with variational inference. In *Master Thesis*, 2018.

[27] M. Simon, S. Milz, Karl Amende, and H. Groß. Complex-yolo: Real-time 3d object detection on point clouds. *ArXiv*, abs/1803.06199, 2018.

[28] Solomon Kullback, Richard Leibler. *Information Theory and Statistics*. Dover Publications, 1959.

[29] Sun, Pei and Kretzschmar, Henrik and Dotiwalla, Xerxes and Chouard, Aurelien and Patnaik, Vijaysai and Tsui, Paul and Guo, James and Zhou, Yin and Chai, Yuning and Caine, Benjamin and others. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2019.

[30] Dustin Tran, Michael W. Dusenberry, M. V. D. Wilk, and Danijar Hafner. Bayesian layers: A module for neural network uncertainty. In *NeurIPS*, 2019.

[31] Dustin Tran, Alp Kucukelbir, Adji B Dieng, Maja Rudolph, Dawen Liang, and David M Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

[32] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger B. Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *ArXiv*, abs/1803.04386, 2018.

[33] Y. Xiang, Wongun Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1903–1911, 2015.

[34] O. Tuzel Y. Zhou. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, June 2018.

[35] Z. Chen. X. Huang. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1856–1860, June 2017.

[36] Kui Jia Zhixin Wang. Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection. *ArXiv,*, 2019.