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

Jaswanth Bandlamudi Hochschule Bonn-Rhein sieg

[jasw](mailto:jaswanth.bandlamudi@smail.inf.h-brs.de)[anth.bandlamudi@smail.inf.h-brs.de](mailto:anth.bandlamudi@smail.inf.h-brs.de)

Anastassia Ku¨stenmacher Hochschule Bonn-Rhein sieg anastassia.ku¨[stenmacher@h-brs.de](mailto:stenmacher@h-brs.de)

Paul G Plo¨ger Hochschule Bonn-Rhein sieg

paul.plo¨[ger@h-brs.de](mailto:ger@h-brs.de)

Luis Octavio Arriaga Camargo DFKI, Bremen [octavio.arriag](mailto:octavio.arriaga@dfki.de)[a@dfki.de](mailto:a@dfki.de)

**Abstract**

*For safe operation of autonomous agents in the world around, an accurate perception and understanding is a must. Object detection is one of the problems in perception stack with an aim of localizing and classifying the object and this task is being accomplished using Deep Neural Net- works with satisfactory results. But, the neural networks are deterministic in nature. Thereby, providing over confident 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 are chose to compare the performances of the Fixed-weight Neural Networks and Bayesian Neural Net- works on the task of localizing and classifying the objects in a 3D world. In the results we show that the fixed-weights neural networks were over-confident in detecting some ob- jects. While, bayesian networks do provide a measure of uncertainty which represents that the model is not certain over the over-confident results given by fixed-weights neu- ral networks.*

## Introduction

Perceiving the outside environment consists of tasks like lane detection, vehicle detection and Vulnerable Road Users (VRU)[1](#_bookmark0) 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](#_bookmark1) around [**?**]. This task has to be carried out with high accuracy and

1Vulnerable Road Users (VRU) are defined in the European Union In- telligent Transportation Systems Directive as ”non-motorised road users, such as pedestrians and cyclists as well as motor-cyclists and persons with disabilities or reduced mobility and orientation”.

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

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 lack the ability to quantify uncertainty in the outputs. The classi- fication models provide predictive probabilities which are softmax outputs of a model. But the softmax probabil- ities 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 au- tonomous driving. Hence, These methods are not compati- ble to safety-critical applications.

Bayesian Neural Networks (BNNs) [**?**, **?**, **?**] are a class

distribution *p*(*w*) and training includes obtaining a posterior distribution over weights represented by *p*(*w D*), where *D* of neural networks in which the weights are initialized as a is the dataset on which the model is trained. *p*(*w D*) cap- tures the model uncertainty. Additionally, BNNs also per-

*|*

*|*

mit the knowledge of uncertainty source [**?**]. The predic- tive uncertainty is classified into epistemic uncertainty and aleatoric uncertainty [**?**]. Aleatoric uncertainty is further classified into Homoscedastic uncertainty, which stays con- stant for different inputs and Heteroscedastic uncertainty

varies with input and some being more affected because of the noise from the environment.

### 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 contribu- tions are three-fold:

Does the Bayesian model perform at the same level of a Fixed-Weight model?

*•*

Can a BNNs version of the 3D object detection be modeled and trained?

*•*

Can the outcomes of the Bayesian model be explained

*•*

?

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

## Related Works

In this section we summarize about various 3D object detection methods and different bayesian neural network ar- chitectures for extracting uncertainty using the chosen 3D object detection network.

### 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 De- tection and Ranging (RADAR), and output a 3D bounding box along with class labels of the agents in the perceived

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 paral- lel network, which consists of an RPN and a detection net- work. 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 in order 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 ob- jects when there is a presence of partial occlusion. The seg- mented point cloud is passed to a PointNet [**?**] based amodal bounding box estimation which regresses the 3D bounding boxes.

### Bayesian Neural Networks

In conventional DNNs, weights are modeled as point estimates which is unjustified. These point estimates re- sulted in the network being deterministic for provided in- puts. Whereas, Bayesian Neural Networks (BNNs) mod- els 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* =

.. Σ . Σ . ΣΣ

*x*(1)*, y*(1) *, x*(2)*, y*(2) *, . . . , x*(*N*)*, y*(*N*) , and *w*

be the weights of the neural network that are to be learnt from the dataset *D* provided. The posterior distribution

*p*(*w D*) is obtained by applying Bayes rule as in Equation

*|*

[1](#_bookmark2)

environment. It is also referred to as amodal 3D object de- tection [**?**], which means the model should fit a 3D bound-

*p*(*w D*) = *p*(*D | w*)*p*(*w*)

*p*(*D*)

*|*

(1)

ing 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 infor- mation 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 informa- tion about height, intensity, and distance and the BEV im- ages hold density, intensity, and height information. Then an RPN generates 3D object proposals from BEV only and

Calculating the denominator *p*(*D*) is not manageable be- cause of the integration over all the weights. The DNNs

generally has millions of weights and integrating all over them is tedious and is computationally intensive. Hence,

we estimate the *p*(*w | D*) distribution.

tion *p*(*w D*) can be approximated to be a parameterized distribution *q*(*w*) and optimize those distributions over the *Graves et al,* [**?**] proposed that the posterior distribu-

*|*

weight parameters to obtain the best approximation.

The approximation is judged using Kullback-Leibler (KL) divergence [**?**] as shown in Equation [2.](#_bookmark3)

KL(*q*(*w*)*ǁp*(*w | D*)) = KL(*q*(*w*)*ǁp*(*w*)) + log *p*(*D*)

*−*E*q*(*w*)[log *p*(*D | w*)] (2)

Hence, VI poses the problem of finding the *p*(*w D*)

*|*

into a maximizing ELBO problem. Once the optimization

problem is defined we can use Bayes by backprop by *Shrid- har 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

With the availability of the approximation *q*(*w*) of *p*(*w*

*D*) obtained using VI method we can infer an output *y∗* for any given input *x∗* by integrating over the weights available

*|*

as shown in Equation [6.](#_bookmark4)

*K*

Σ . Σ

replacing the weights with a parameterized distribution. Hence, the optimal parameters *θopt* which consists of re- spective mean and variance (*µ, σ*) are defined as

*K*

*p* (*y∗ | x∗, D*) *≈* 1

*p y∗ | x∗, wk* (6)

*k*=1

*θopt*

= argmin*θ* **KL**[*q*(**w** *| θ*)*ǁP* (**w**)]*−*

E*q*(**w***|θ*)[log *P* (D *|* **w**)]

(3)

## Epistemic Uncertainty Quantification

In this section we will summarize the method to quantify

Substituting this in Equation [3](#_bookmark5) we get,

*θ∗* = argmin*θ* E*q*(**w***|θ*)*f* (**w***, θ*) (4) But, the above object function cannot be used with a

standard back-propagation algorithm. This is due to the inability to achieve a differentiation of the objective func- tion. Hence, *Kingma et al* [**?**] proposed a reparameteriza- tion trick in which the gradients are calculated through gen- eral back-propagation and also reducing the variance. In case of CNNs, *Shridhar et al.* proposed to sample layer

the uncertainty in 3D object detection.

### Quantifying Epistemic Uncertainty in Classifi- cation

Uncertainty in classification can be quantified using the Shannon entropy of the final softmax scores in Equation [8.](#_bookmark7) As used by *Feng et al.* [**?**]. For an input of *x∗* the clas- sification probability of class for N with the availability of the probability score is calculated using Equation [7.](#_bookmark6) This probability is used to calculate Shannon entropy (SE) using Equation [8](#_bookmark7)

activations instead of weights, it results in a estimated pos-

terior distribution *qθ* (*θ*i*jhw Data*).The resultant equation

*|*

for sampling the activation is written as

*p* (*c* *|* **x***∗*) = E

*p*(**W***|***X***,***Y**) *N*

*p* (*c |* **x***∗,* **W**)

(7)

*≈* 1 Σ *si*

*∗***x**

. 2 2

*bj* = *A*i *∗ µ*i + *εj ·*

*A*i *∗* (*α*i *· µ*i ) (5)

*N*

*i*=1

*b* is the layer activation, *εjsN* (0*,* 1), *A* is the receptive

field area.

*SE* (**y***∗ |* **x***∗*) = H (**y***∗ |* **x***∗*)

= *−p* (*c |* **x***∗*) log *p* (*c |* **x***∗*) *− p* (*¬c |* **x***∗*) log *p* (*¬c |* **x***∗*)

.

.

Σ

*Flipout based Weight-sampling methods:* In order to per-

Σ

*≈ −* 1 Σ*N*

*N*

*si ·* log 1 Σ*N*

**x**

*si ∗ −*

**x**

*N*

**x**

form inference for a new input *x∗* given a VI approximation

*q*(*w*) there is a need to sample the weights from the approx-

imation for every inference run performed in order to ex-

*N*

1 *−* 1

*i*=1

*N*

Σ

*i*=1

**x**

*si ∗*

*N*

log

*i*=1

1 *−* 1

*N*

*i*=1

*si ∗* Σ

(8)

tract *p* (*y∗ x∗, w*).using Flipout [**?**] introduced in *Wen et*

*|*

*al.*. This method aims to uncorrelate the gradients between

every sample. This work achieved by making some assump- tions which are

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

Flipout suggests to use a posterior distribution (*qθ*) which is jointly used by each and 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 *qθ*. The weight sampling is done by multiplying *E* and the posterior distribution

If the SE value is zero, then the network is most certain

of the output and when SE value is higher the network is

not certain of the classification result.

### Quantifying Epistemic Uncertainty in Regres- sion

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

* + 1. Transform the bounding box detections into world ref- erence frame and the mean value of the detections re- gressed during N forward passes is calculated.
    2. Covariance matrix of the detections is calcu-

lated using the following the equation *C* (**x***∗*) =

1 Σ*N i iT T*

*W* = *qθ, E*

*N*

*i*=1 **v**ˆ**x***∗* **v**ˆ**x***∗ −* **Ix***∗* **Ix***∗* .

* + 1. Total variance is calculating by taking the trace of the

. covariance matric *C*(*x∗*).

The Total variance spans from [0*,* + ), the higher the value represents higher the epistemic uncertainty in re-

*∞*

gressed bounding box parameters.

## Experimental Results

In this sections we explain various experiments, their re- sults and comparing the performance of different networks.

### 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 sam- ples of batch size 32. The training is done for 178 (i,e.

bounding box center and rotation angle to calculate the epis- temic 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 |

Three days) epochs.

latex/images/Part-Bayesian\_F\_Poin

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 |

Figure 1. Entropy for Car detection

latex/images/Part-B

F*P ointnetResults/EntropyCyclist.pdf*

Figure 2. Entropy for Cyclist detection

### Bayesian Neural network modelling and evalu- ation

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,

F*P ointnetResults/EntropyP edestrian.pdf*

Figure 3. Entropy for Pedestrian detection

### Visualizations

latex/images/Part

by fixing the weights of instance segmentation model and converting the spatial-transformer network and bounding box network into Bayesian model which fitted during train- ing.

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 distri-

bution *p*(*w*). The resultant models hold different sampled

weights and does form an ensemble like architecture and

helps in extracting epistemic uncertainty. The epistemic un- certainty should be quantified using Shannon entropy and Total variance. We used the bounding box dimensions,

#### 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.](#_bookmark8) Num- ber 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.

#### Epistemic uncertainty due to cluttered environ- ments

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

#### 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

# latex/images

Figure 4. Predictions in image plane. Figure 6. Predictions in BEV projection of LiDAR point cloud.

# latex/images

Figure 5. Predictions in LiDAR plane.

* + - * 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

*•*

## Conclusion and Future works

In this research and development project, Frustum- PointNet [**?**] is modelled 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 same level

Figure 7. Predictions in image plane.

as the network trained using KITTI dataset [**?**]. The com- parison is done based on 3D Average-Precision values cal- culated using fixed limits set on Intersection-over-Union values.

We also modelled a Bayesian Neural Network based on Frustum-PointNet [**?**] to extract the uncertainty due to the weights in the Frustum-PointNet [**?**] model. We also con- cluded 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 better tuning of the prior distribution placed on the weights, we can achieve better results.

Further analysis on the extracted uncertainty also showed

# latex/images

Figure 8. Predictions in LiDAR plane.

Figure 10. Predictions in image plane.

# latex/images

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

that the model is over-confident over some samples. We also visualized some samples which represented the mod- els over-confidence and confirm the need for the usage of uncertainty as a measure of confidence.

Figure 11. Predictions in LiDAR plane.

# latex/images

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