Uncertainty Aware 3D Object Detection

M.S. Şimşek

Delft University of Technology

TU Delft

Abstract

Object detection is currently one of the most prominent topics and is a fundamental concept that will shape the future of autonomous driving. Hundreds of papers are being published every day in an attempt to enhance the performance of these detectors by creating more complex models, albeit with marginal improvements. The issue with the current state-of-the-art approaches lies in the lack of quantifying the uncertainties associated with the model and the datasets. This represents a significant research gap as these uncertainties are neither considered nor integrated into the model itself. This research paper addresses the question of how to quantify and utilize the uncertainties of a 3D LiDAR-based object detector. The paper demonstrates that in the world of semantic segmentation and 3D object detection, multi-frame detectors hold great potential for incorporating uncertainties to not only enhance model performance but also improve uncertainty quantification.A key contribution of this paper is the revelation that the most effective technique for quantifying uncertainties is to combine Monte Carlo dropout and assumed density filtering. This combination approximates the uncertainties, which can then be used to represent the spatial uncertainties associated with detected objects. This research underscores the absence of spatial uncertainties for multi-frame detectors in existing literature. Furthermore, to develop such a model, the study conducted a comprehensive analysis of fundamental existing models and datasets.

Contents

1 Introduction

The growing interest in autonomous driving has led to increased pressure for advancements in 3D object detection in the field of computer vision and perception. Besides future fully self-driving functions, there are already many advanced drivers assistance systems (ADAS) that not only aid the driver during specific manoeuvres (Parking Assist, Adaptive Cruise Control), but also act as an emergency to prevent road accidents (Pedestrian Avoidance, Automatic Emergency Braking). 3D object detection is a crucial part of ADAS system as it involves detecting and locating objects in a 3-dimensional environment, providing a deeper understanding and representation of the surrounding world. For the detection of the 3D surroundings LiDAR sensors are used. LiDARs (**Li**ght **D**etection **a**nd **R**anging) are sensors which can detect the distance of an object, based on light pulses which are emitted on to on an object and reflected back. This 3D representation of the environment can then be used to detect certain objects, e.g. pedestrians, cars, road signs etc.
Over 8,500 papers have been published on the topic of 3D object detection in the past two years, indicating a significant level of focus and attention on the subject [1]. Most object detection papers concentrate solely on the performances of the deterministic predictions, neglecting the uncertainties of the predictions [2–5]. This approach disregards the words of Bertrand Russell, a renowned mathematician and philosopher, who stated in 1929 that ”Probability is the most important concept in modern science, despite the fact that nobody has a clear understanding of it.” This mindset is prevalent in the current AI and object detection landscape, where authors strive to incrementally improve existing models without considering the potential uncertainties that arise. It is crucial for other departments such as path planning and motion planning to comprehend the uncertainties in the productions, as the development of self-driving vehicles is a multi-disciplinary project.

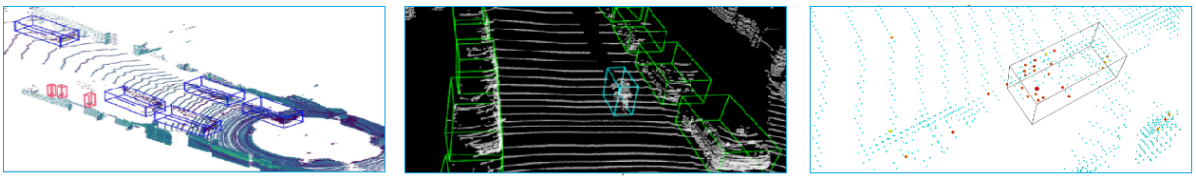

Figure 1.1: Examples of the outputs of three pioneering LiDAR based object detections (Point Pillars, VoxelNet and Voxel Transformer) where the outputs are shown in a deterministic manner

Uncertainties should be comprehensively quantified whenever we are dealing with predictions. In the field of object detection, there are two main approaches: single-frame object detection and multi-frame object detection. Single-frame object detection involves identifying objects based solely on data obtained from a single frame of a LiDAR sensor. Conversely, multi-frame object detection considers the current frame, as well as previous and future frames, to enhance the accuracy of object detection. This latter approach is particularly useful in offline processes, such as the evaluation of advanced driver-assistance system (ADAS) functions.
The object detection models can be categorized into three modules. First, there is a module for pre-

processing the LiDAR data. From this preprocessed data, features are extracted, often referred to as a feature extraction module. These features are then finally fed into the region proposal network which utilizes these features to predict the bounding boxes.

In this paper we devise a novel neural network called Uncertainty-Aware LSTM that uses concatenated LiDAR frames to predict the semantic scores and their uncertainties. Subsequently, the point clouds and semantic scores gets fed into an object detection module head which predicts the objects position, size and orientation. Finally from the approximated quantities of the previous two modules, a spatial uncertainty detection module will give us the uncertainty represented in a probabilistic distributed bounding box form. The main research question of this research is: **How can the uncertainties of a multi-frame 3D object detector be quantified and used in 3D object detectors?**

To do that, the following three subquestions will be answered:

- How can we predict the semantic scores with their uncertainties by using multiple frames
- How can we predict the bounding boxes with their uncertainties with the use of multiple frames
- How can we quantify the spatial uncertainty of the predicted objects

2

Related Work

This chapter provides an overview of uncertainty-aware object detectors, examining state-of-the-art approaches and exploring the various types of fundamental uncertainty detection in neural networks.

Object detections
Before jumping to uncertainties, it is crucial to gain an understanding of the related work in object detection and how single-frame detectors differ from multi-frame detectors in processing aleatoric and epistemic uncertainties in neural networks. In the context of processing 3D point clouds for object detection, previous works have employed either a point-wise, voxel-wise, or a combination-based approach. For instance, in [2] a voxel grid is used to divide the point cloud into voxels and encode it with a Voxel Feature Encoding (VFE) layer. The resulting features are then processed by a Convolutional Neural Network (CNN) and a Region Proposal Network (RPN) to generate detections. Another work, [6], employs a transformer-based architecture that leverages self-attention to capture long-range relationships between voxels. Multi-frame object detection represents an extensive network that leverages multiple frames to predict objects. There are only a handful of papers which make use of this type of object detection. In [7] a 3D multi-frame attention network is used where a novel fast single-frame detector is used to predict box proposals. These box proposals are then stored in a memory bank and used in future frames to help extract object sizes and positions. In [8] a multi-view frame detector is used to predict the 3D size of vehicles. The paper showcases that their performance greatly surpasses many of the state-of-the-art single-frame object detectors, such as PointPillars.

In [9] the authors use a similar approach as in [7] to detect objects in point clouds. It uses a sparse convolutional single-stage predictor which extracts voxel features from the raw point clouds. This concatenated data from the previous frames and the current frame gets passed into an LSTM model (explained in section 3.2.2). The LSTM model serves as an effective tool for handling sequences and retaining important features in memory. Object detection is subsequently performed using a straightforward encoder-decoder technique to extract the detected objects.

The main issue with multi-frame object detectors is that they tend to become very complex, as noted in [10]. This complexity is mainly due to the fact that we have to consider multiple point clouds, as opposed to just one. However, despite this complexity, these methods can achieve great success in terms of performance.
Modelling uncertainties in object detection
Modelling uncertainties in object detection based on neural networks is an important feature for assessing the robustness of object detectors. While there are only a limited number of papers that employ the same approach to estimate prediction uncertainties, several papers discuss techniques for estimating uncertainties. In , the technique of using dropout as a Bayesian approximation has been introduced to represent model uncertainties in deep learning. Despite its widespread use as a regularization technique, Bayesian dropout is not suitable for certain network architecture as they can introduce instability in Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) . Furthermore, other popular techniques to quantify epistemic uncertainties is Monte Carlo dropout [11]. Variational

inference is a approximation technique used in neural networks, where it considers the Bayesian inference problem as an optimization problem [12].

In [13] the authors capture the aleatoric and epistemic uncertainties using the MC dropout method and proved that by incorporating these uncertainties in the object detection module, the performance was increased by 1-5%. The bounding box spatial uncertainty is captured by using the total variance of the covariance matrix of the N forwards pass regressions. This way, the 8 corners of the bounding box get a certain epistemic and aleatoric variance. Unlike [14] this paper incorporates the aleatoric uncertainties and the epistemic uncertainties in the final bounding box distribution. However, it does not further explain the distribution of the predicted bounding box and a high quality end-to-end uncertainty detection model, where the uncertainties are quantified and processed throughout the whole network is missing.

To summarize, this paper establishes connections between the components of object detection, uncertainty propagation in neural networks, and spatial uncertainties. While previous research has primarily focused on individual instance detection and quantification, our work aims to integrate these aspects to create a comprehensive model for object detection from 3D LiDAR data. We investigate how uncertainties are propagated throughout the network and explore the potential of utilizing these uncertainties as valuable features in the final product.

3

Methodology

This chapter discusses the methodology used to address the research questions introduced in the first chapter. Chapter 3.1 discusses the feature-extracting module. Chapter 3.2 describes how these extracted features are fed into an object detection module. Finally, Chapter 3.3 explains how the final uncertainty is quantified.

3.1. Problem description

The general problem that arises for this research is to find a model which takes in a sequence of LiDAR data and gives us bounding boxes of the detected objects, along with their spatial uncertainties where the uncertainties are extracted from the sensor and model uncertainties. The first module where the point cloud gets send to is the feature extracting module.

3.2. Feature Extracting Module

The feature extracting module is a module which predicts the semantic scores of the LiDAR points. Given an unordered set of point cloud P_t at time t, the pointcloud is defined as:

$$\boldsymbol{P} = \{(x_i, y_i, z_i, r_i) \quad | \quad i \in N\} \tag{3.1}$$

Since we Let N be the number of points in the point cloud and suppose that there are n different classes in the point cloud. The semantically segmented points $\boldsymbol{C} = \{C_i \mid i \in N\}$ are a set of classes where each variable C_i is taking a value from the set of possible classes $\{1, 2, ..., n\}$. Semantic segmentation for 3D LiDAR point clouds can be divided into three groups: point-wise, voxel-wise or point-voxel wise. In [15] it has been showcased that the point-voxel wise methods provide the advantages of the pointwise and voxelwise methods. By using the point based methods we are able to keep relationships between points, while also having the advantage of the voxel wised regularity models.

The Feature extracting module consists of two parts: the single-frame detector and the LSTM module.

3.2.1. Single Frame Detector

The single-frame detector is a Sparse-Point Voxel model that incorporates aleatoric and epistemic uncertainties to predict the semantic scores of a single point cloud frame. It consists of two parts: the voxel-wise method and the point-wise method.

Sparse-Voxel Network

The voxel-wise method uses a sparse point-voxel method where a Sparse Convolutional U-net is the main component of the model. The raw point cloud first gets voxelized into a fixed grid. The voxel features are then fed through the Sparse Voxel U-net with skip connections. However, different from the state-of-the art U-net models, the aleatoric uncertainties of the LiDAR system are also taking into account in this network by adding Assumed Density Filtering (ADF). Assumed Density Filtering is a

technique used in Bayesian Neural Network where the aleatoric uncertainty are assumed to be known by the sensor's noise characteristics. These are often given in the sensors data sheet. These aleatoric uncertainties can thus be expressed as a normal distribution $\mathcal{N}(r, v)$, where 'r' represents the sensor's reflectivities, and 'v' represents the known sensor noise. Consequently, activation functions are replaced by probability distributions, enabling the quantification of aleatoric uncertainties denoted as σ_{al}. Finally, the output is devoxelized to return the semantic scores to their point-wise format.

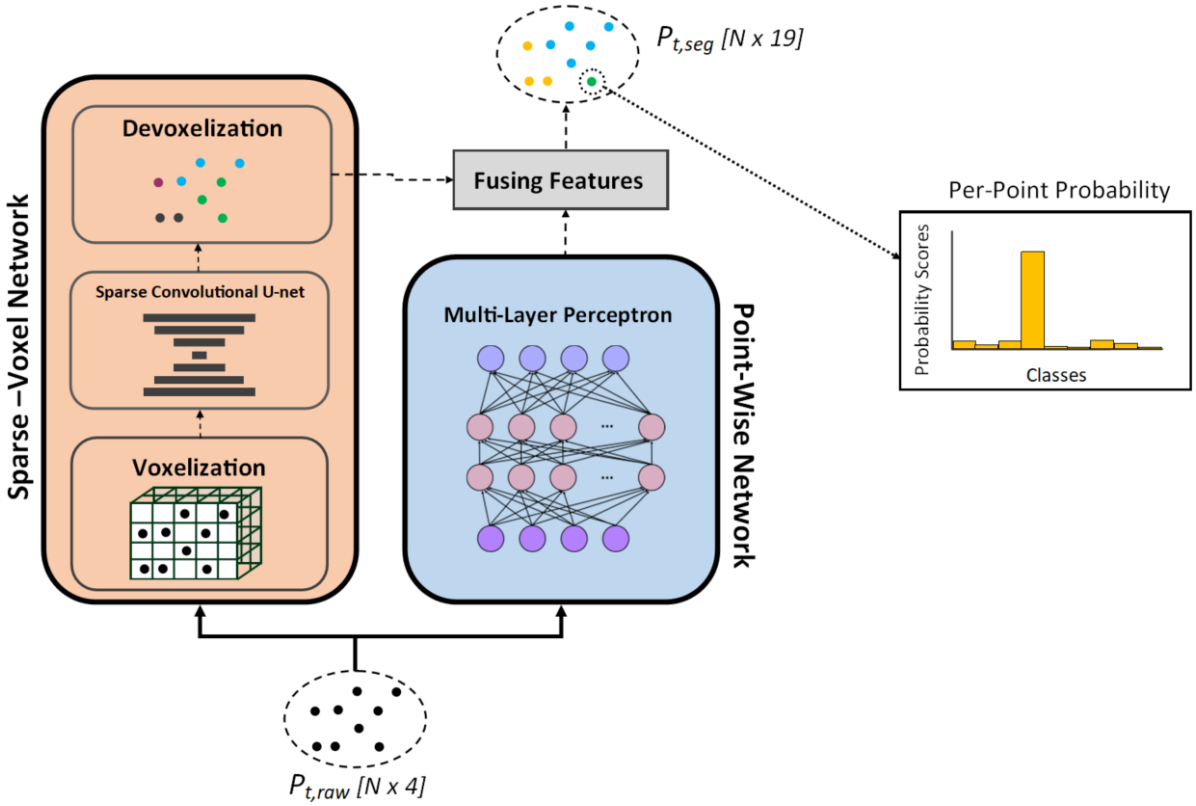

Figure 3.1: Image showing the single frame detector network. On the left the Sparse-Voxel Network takes the voxelized input and applies convolution to extract the coarse-grained voxel information. The Point-Wise network on the right consists of a MLP in order to keep the point-level information. The probabilities are added together from both network to output the semantic scores of the point cloud.

Point-Wise Network

The Point-Wise Network is a lightweight Multi-Layer Perceptron that considers all points in the network. The MLP is used to extract individual point features for each point, directly addressing each point to obtain distinct and discriminative features. Despite its simplicity, the MLP is effective at capturing high-resolution information for individual points. This fine-grained, point-level information is crucial in complementing the coarse-grained voxel-based information. The points are scored in the same semantic classes as the Sparse-Voxel Network.

Fusing Features

Since the Sparse-Voxel Network and the Point-Wise network share the same format (N points, 19 class scores), their features can be easily fused together by adding the outputs from both networks. However, the outputs are still defined as a probabilistic output with a mean and a variance. The Gaussian probability density is calculated to obtain a single probability per point per class, resulting in the desired output format.

3.2.2. LSTM Module

The LSTM module is the next stage of the model, where the outputs of the single-frame detector and the outputs of the previous frames are sent to. The LSTM module is there to take advantage of the detections in our previous frames. It consists of three components: the joint voxelization module, the

LSTM cell, and the joint devoxelization module.

Joint voxelization

Joint voxelization is a technique in which we voxelize the point clouds based on two point cloud frames. The semantic scores from the previous frames are jointly voxelized with the semantic segmentation scores from the previous frame. The hidden features of the LSTM cell from the previous timestamp ($t-1$) are used to do this. Before transforming the point cloud to the current timestamp, the point cloud with semantic segmentation is first passed through a score filter where only the points containing interesting features are selected, and non-important points are filtered out. This is done to improve the model's efficiency, as some features (e.g., grass, road plane, etc.) do not need to be fine-tuned by the LSTM. After selecting only the interesting points from the score filter, the point cloud P_{t-1} is transformed into the current frame using *ego motion transformation*. Unlike ego motion compensation, ego motion transformation occurs between two timestamps, whereas ego motion compensation deals with the rotation during a single timestamp. Assuming that the transformation matrix T, equation 3.3 describes how every point $i \cap N$ gets rotated and translated from timestamp $t-1$ to t:

$$\boldsymbol{P}_{tf} = \boldsymbol{P}_{t-1}\boldsymbol{T}^T \tag{3.2}$$

Where:

$$T = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & d_x \\ sin(\theta) & cos(\theta) & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \tag{3.3}$$

and

$$\boldsymbol{P}_{t-1} = \begin{bmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_i & y_i & z_i & 1 \end{bmatrix} \tag{3.4}$$

In this calculation, we assume that the vehicle only rotates in the yaw direction, neglecting any roll and pitch movements. Now that the point cloud is transformed into the current frame, the output of the single-frame detector and the transformed point cloud can be jointly voxelized. We use joint voxelization to account for the differences between the points of the transformed point cloud and the point cloud coming from the single-frame detector. Since the transformed point cloud has errors due to the movement of objects between frames t-1 and t, joint voxelization ensures that those points are not discarded but can still be used in future sparse convolutions. Instead of discarding these points, they are given a zero value. The pseudocode for joint voxelization is as follows:

Algorithm 1 Joint Voxelization

$N_{tf} \leftarrow n_{tf}$
$N_{current} \leftarrow n_{current}$
$P_{final} = (voxels = [], features = [])$
while $n_{tf} \neq N_{tf}$ **do**
  if $n_{tf} \cap N_{current}$ **then**
    $P_{final}[voxels]$.append(n_{tf})
    $P_{final}[features]$.append$(\textbf{\textit{Average}}(feat_{tf}, feat_{current}))$
  else
    $P_{final}[voxels]$.append(n_{tf})
    $P_{final}[features]$.append(0)

In theory, joint voxelization makes sure that none of the important points are being discarded in the sparse convolutions by adding a non empty value in the voxel which does not overlap between the two point clouds. Finally, we will have a jointly voxelized point cloud where the points and semantic scores

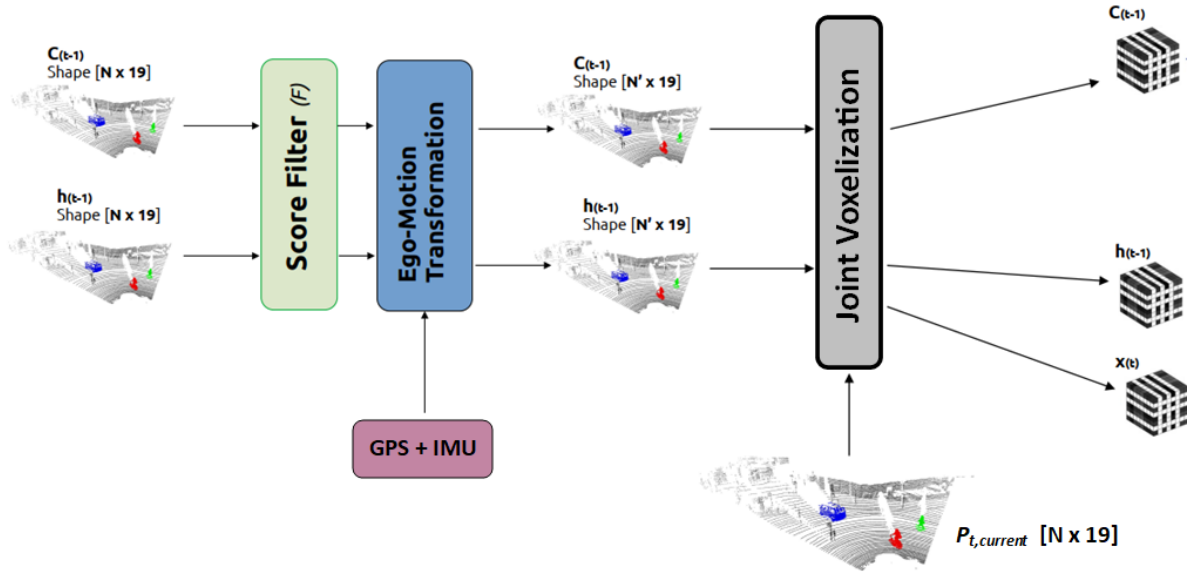

Figure 3.2: Figure showing how the output of the LSTM's previous frame are jointly voxelized with the current single frame detector predictions in order to keep relevant data in the loop.

coming from the previous frames are fused together with the output of the single frame detector. The voxels and features originating from the previous timestamp are fed along with the output of the single frame detector to the LSTM cell.

Vanilla LSTM Cell

To find the relationships between previous frames and the current frame, an adapted vanilla LSTM module is used. LSTMs are an efficient type of recurrent neural network that excel at handling sequential data and retaining important features in memory. A vanilla LSTM cell consists of an input gate, an output gate, and a forget gate. In addition to these gates, the LSTM also includes two states: the cell state and the hidden state. The cell state is the memory of the LSTM cell and stores information about all computations from the previous timestamp. It can be considered the 'long-term memory' of the LSTM. The hidden state is the output from only the previous timestamp and can be considered the 'short-term memory.' The forget gate controls how much previous knowledge of the LSTM should be forgotten and how much information should be retained. The input gate determines how much of the input information and which information should be added to the cell state from the new inputs. Using the results from the input and forget gate, the previous cell state C_{t-1} is updated to form the new cell state C_t. The final updated hidden state is calculated by passing the latest cell state C_t through a tanh activation function [16] and multiplying it by the results of the output gate. The LSTM forward pass for one operation can be mathematically described as:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{c_t} &= tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ c_t &= f_t \times c_{t-1} + i_t \times \tilde{c_t} \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \times tanh(c_t) \end{aligned} \tag{3.5}$$

Adapted LSTM

Since the input to the LSTM module is a voxelized pointcloud containing the aleatoric uncertainties, the LSTM module is changed in order to be able to handle this kind of data. For starters, the fully connected layer of the LSTM in the input is replaced by a lightweight sparse voxel network. This network will give us the candidates for the cell state updates and the other LSTM gates. The advantage for this kind of

process is that we can re-use the network from the single frame detector, only removing the pointbased side of the network. This network is able to concatenate the multiframed voxelized point clouds and pass it through to the before mentioned cell candidates.

Furthermore, instead of the vanilla LSTM cell, a newly designed Monte-Carlo type of LSTM cell has been developed. Monte-Carlo dropout is a dropout technique used to quantify aleatoric and epistemic uncertainties. By doing this, we ensure that every model's uncertainties are taken into consideration when the point cloud data is propagated through the network. The Monte-Carlo LSTM cell is built as follows: We first train our model with a plain vanilla LSTM layer and save the weights of all the layers in our sequential model. From there, we rebuild a new sequential LSTM layer where the weights are reloaded from the plain LSTM network. Finally, Monte-Carlo dropout is applied to extract the class uncertainties from the model.
The updated memory and hidden features are still represented in a jointly voxelized set. Before this batch is send to the next frame, the point clouds first need to be devoxelized.

Joint Devoxelization

In joint devoxelization, we make sure that the jointly voxelized point clouds are returned to their point representation. The process is the same as for joint voxelization, but in a reversed order. This makes the data usable for the next LSTM cell state. The devoxelized memory features contain the highest quality semantic scores from the current timestamps point cloud.

3.2.3. Object Detection

The devoxelized memory features coming from the LSTM cell, which contains the highest quality probabilistic semantic scores, are fed through the final object detection module. The object detection module consists of a sparse convolutional encoder-decoder network. Assumed density filtering is used in the network in order to preserve its uncertainty detections. To do this, the normal distributions in the activation function will be transferred from the previous network.

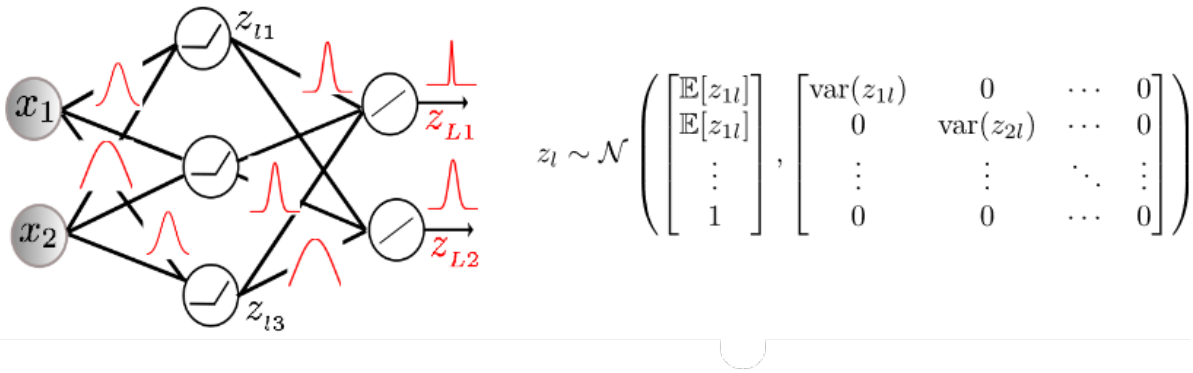

Figure 3.3: Image showing how assumed density filtering is used in the network's activation functions, in order to quantify the uncertainties of the model.

To find object detections from the features of the point cloud, we will use a per-point object detector. This means that instead of having a fixed number of bounding boxes to be detected, we detect a bounding box for every point in the network. This allows every point to be considered in the network, and instead of having a small set of bounding boxes, we can have N number of bounding boxes. This is especially useful for scenarios where the scenery can vary between different frames. The object detection module predicts the box centers (x, y, z), box sizes (length, width, height), a 3x3 rotation matrix specifying the orientation of the objects with respect to the car's frame, and a 2D box score. The two-dimensional box scores provide us with the score probability and classification probability for each bounding box. The score probability is extracted based on the dimensions of the bounding boxes, and the class probabilities come directly from the per-point probabilities, where the probabilities of all the

points are weighted and fused together.

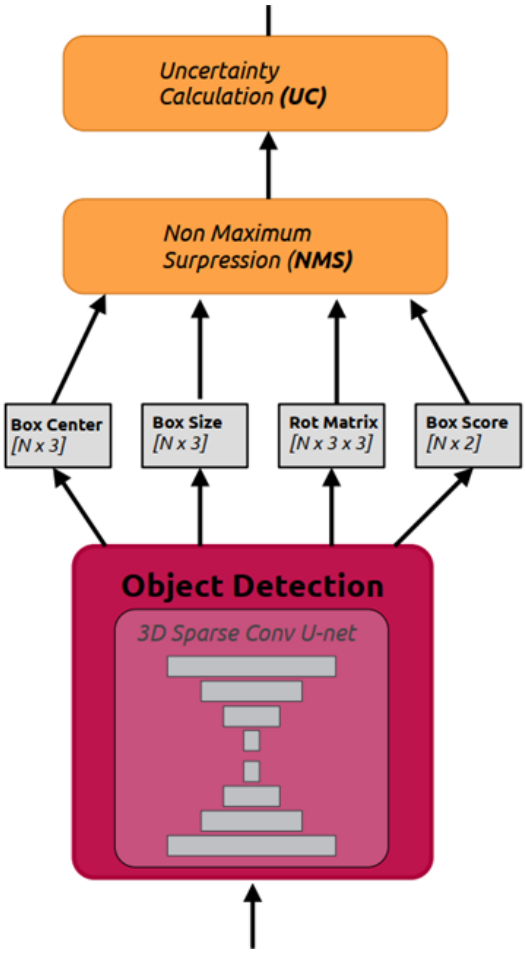

Figure 3.4: Image showcasing the architecture of the object detection module, where it consists of three parts. The first part gives us a per point bounding box prediction. The second part applies Non Maximum Surpression in order to reduce the number of bounding boxes from N to N'. The final part extracts the spatial uncertainties of the bounding boxes.

Non Maximum Surpression

To reduce the large number of proposal bounding boxes predicted by the object detector to only the high quality bounding boxes, Non Maximum Surpression (NMS) is applied [17]. NMS is a state-of-the art technique to filter the proposed bounding boxes by iterating through the proposals, computing the Intersection over Union (IOU) of this proposal with every other proposal and check if it exceeds a certain threshold N. This is repeated until there are no more proposals left in the field.

Uncertainty Calculation

Now that only a handful bounding boxes are selected from the NMS, the final module calculates from the uncertainties, the spatial uncertainty of the bounding box. The idea of probabilistic box representation is proposed in PDQ for 2D axis-aligned bounding boxes of images [18]. This same principle has been transferred to 3D aligned bounding boxes, where a natural generalization of the spatial distribution $P(u) \subset [0, 1]$ has been transferred to a 3D space. The resulting spatial distribution in Bird's eye view for a 3D rotated bounding box $B(y)$ is given by:

$$P_{PDQ}(u) = \int_{y|u \subset B(y)} P_{\hat{Y}}(y|x)dy \tag{3.6}$$

Here, $P(u)$ is the probability that u is a point of the object. Equation 3.6 is easy to calculate for axis-aligned bounding box detections, but fails to mitigrate the spatial distributions for 3D rotated bounding boxes, since it has to integrate over the space of y, which is 7 dimensional for 3D. To solve this, a

transformation in the integral will give us another expression as a probabilistic density function (PDF):

$$P_G(u) := \int_{v_0 \subset B(y)} P_{V(v_0,\hat{Y})}(u) dv_0 \tag{3.7}$$

Here, v_0 and $\hat{Y}$ describe the universal affine transformation between the bounding box and the point. Since the point uncertainties in the equation originate from the neural network model, it is unnecessary to investigate the noise of every point per timeframe, as this is already handled by the LSTM network. The probability distribution assumes that the uncertainties have already been propagated through the network. Now that the spatial distribution of every point has been calculated, we will only select the points that have a high enough probability of belonging to the bounding box. In this consideration, only the edges of the bounding boxes are used for their spatial uncertainty.

Now that we have a filtered set of points along with their spatial distribution, we assign each of these points to the corner of the bounding box closest to the point. By using a weighted average, we aggregate the probability distributions of these points into a single probability distribution. By linearly interpolating the distances between these probabilities and the corners, we are finally able to calculate a spatial uncertainty for these bounding boxes (see Figure 3.5).

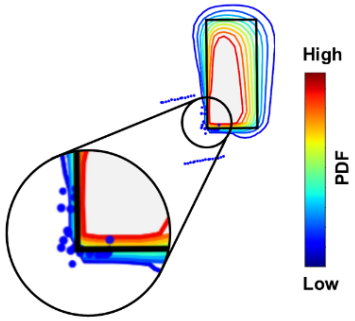

Figure 3.5: This figure showcases an example of a PDF distribution for a predicted bounding box of a vehicle. As can be seen from the image, the lower left corner of the image has a more narrow distribution while the top of the car (which contains the least amount of points has a very wide probability distribution.

Jaccard Intersection over Union

IoU is one of the most commonly used metric for detection. It has an intuitive geometry definition measuring the overlap between the predicted and ground truth bounding boxes. Despite its popularity, IoU only applies to deter-ministic predictions and labels. In this section, we define a metric over p(u) of probabilistic bounding box following the definition of the probabilistic Jaccard index **??**.

$$JIoU := \int_{R_1 \cap R_2} \frac{du}{\int_{R_1 \cup R_2} \max\left(\frac{p_1(v)}{p_1(u)}, \frac{p_2(v)}{p_2(u)}\right) dv},$$

where p1, p2 are the spatial distributions of two boxes, as introduced in equation **??**. u, v are points in the 3D or BEV space. R1, R2 are the supports of p1, p2, respectively. Note that JIoU degenerates to IoU when two boxes are deterministic, i.e., p(y|x) is a delta function, where R1, R2 become bounding boxes and p1, p2 become uniform inside their boxes.

4

Results

In this work, we propose (1) a generative model for detecting objects in 3D LiDAR point clouds, (2) an explanation of how both aleatoric and epistemic uncertainties propagate through the model, and (3) a method for visualizing these uncertainties through a spatial distribution.

First, we will outline our choices regarding the neural networks and the types of uncertainty detection methods used to construct the network.

Second, we will justify our methods and demonstrate that the spatial uncertainty calculation aligns with experiments conducted on existing datasets

4.1. Experiment Details

Single-Frame Detector The voxel size used in the frame detector is set to 0.05m, based on a 100m x 100m x 10m voxel grid. The sparse convolutions on the sparse tensors are derived from the architecture of [19]. The 3D sparse U-net comprises three encoder blocks (256, 128 and 64), one bottleneck block, and three decoder blocks (64, 128 and 256). The MLP for extracting point-wise features consists of two layers, with dropout applied to reduce overfitting of the point features.

LSTM Network
The lightweight sparse voxel network in the LSTM cell has the same architecture as the Single-frame detector, where the layers to improve computational efficiency. The Sparse convolutional network consists of a single encoder (128 dimensions), one bottleneck and one decoder layer (128 dimensions) with max pooling in between. Also, the voxel size has been doubled to 0.10m. The adapted Vanilla LSTM module consists of two layers, with Monte Carlo Dropout applied in between these layers. This dropout technique helps to introduce stochasticity and regularize the model during training, ultimately improving its generalization performance.Before commencing training, the weights of the LSTM module were initialized. Proper weight initialization is crucial in neural network training to prevent issues such as vanishing or exploding gradients and over-saturation of activations.

Object Detection module
The object detection module used consists of a 3 layer encoder decoder network. The dimensions are 256, 128 and 64 for the encoder and in reverse for the decoder network. In between, a bottleneck layer connects the decoder with the encoder. In the end, a fully connected layer with 14 outputs have been used. which helps us to create a per point object detection prediction. An overlap threshold of 0.4 has been used to select the number of bounding boxes.

Training
The single frame detector is first trained seperately based on the semanticKITTI dataset, with a linearly scheduled learning rate starting from 0.05 up to 0.001. The multi-frame semantic segmentation module have been trained for models having single-frame detections, 2,3,4 and 5-framed detections. The object detection module has been trained on the Kitti dataset where the ground truth semantic

segmentations from the dataset have been used instead of the one from the multi-frame detections.

4.2. Experiment Results

Several metrics have been used to evaluate the performance of the network. Since the networks can be considered as a seperate module, they will all be evaluated in a seperate form of manner.

Semantic Segmentation

The dataset used to evaluate the performance of the semantic segmentation module is the test set of SemanticKITTI. The mean intersection over union to evaluate the performance of the semantic segmentation. In order to find the best model, the number of frames used for the LSTM model is compared. The sequences 0-8 of the semanticKITTI dataset have been used for training, while the sequences 9-10 have been used as our test set, giving it a solid distribution of the train and test set. The results are given in the table below.

Number of frames	mIoU
1-frame	57.2
2-frames	**66.6**
3-frames	64.1
4-frames	53.9
5-frames	48.7

Table 4.1: Results for the semantic segmentation module comparing number of frames used in the LSTM model.

Based on the results of the frame-by-frame comparison in the semantic segmentation module, it is evident that the best performance is achieved when using a 2-frame setup for the LSTM model. The 2-frame model surpasses other state-of-the-art single-frame semantic segmentation models on the semanticKITTI dataset.

Model	mIoU
PointNet	14.6
SPGraph	17.4
PointNet++	20.1
KPConv	58.8
MinkowskiNet	63.1
UA-LSTM (2-frames)	**66.4**

Table 4.2: Comparison of our 2-framed network against other state-of-the-art semantic segmentation models, tested on the semanticKITTI dataset.

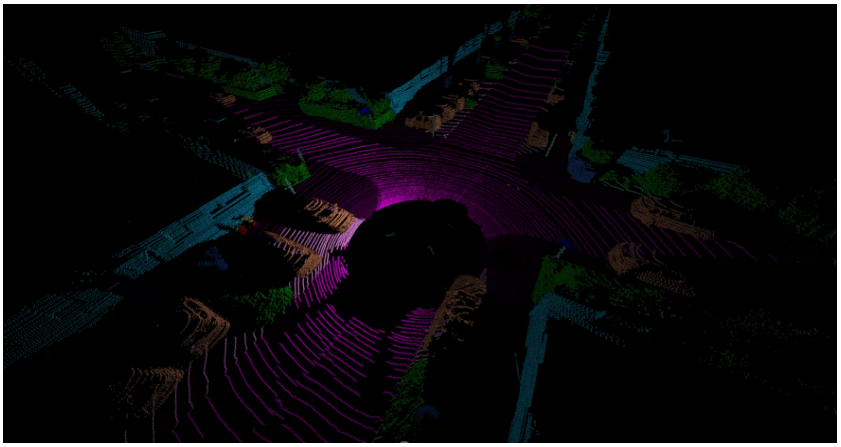

Figure 4.1: Results of the two-framed semantic segmentation model

3D Object detection

The dataset used to evaluate the performance of the 3D object detection module is the test set of the KITTI dataset. The mean average precision (mAP) has been employed to assess the model's performance. Since our 3D object detection model necessitates semantic scores as input, the KITTI test dataset was initially processed through the semantic segmentation module, with the number of frames adjusted accordingly. The evaluation has been conducted for the car, pedestrian, and bicycle labels. The results of the frame-by-frame comparison are presented in the table below:

Number of frames	Car	Pedestrian	Bicyclist
1-frame	79.1	61.1	46.6
2-frames	**88.3**	**65.2**	**49.6**
3-frames	87.7	65.0	44.1
4-frames	80.4	59.6	44.0
5-frames	72.7	46.7	38.2

Table 4.3: A table comparing the results of the frame comparison on the 3D object detection module. The models are evaluated using the KITTI test dataset, with the mAP values reported in the table.

From the results, it is evident that the 2-frames model performs the best for cars, pedestrians, and bicyclists. When comparing the results of the 3D object detection module to other state-of-the-art models, it becomes clear that the model outperforms them. The results are presented in the table below:

Model	Car	Pedestrian	Bicyclist
BirdNet	76.15	41.55	44.00
PS++	74.64	36.00	44.45
CG-Stereo	74.39	33.22	47.40
HybridPillars (SSD)	86.22	44.81	76.32
UA-LSTM (2-frames)	**88.3**	**65.2**	**49.6**

Table 4.4: Table comparing state-of-the-art 3D object detectors to our model.

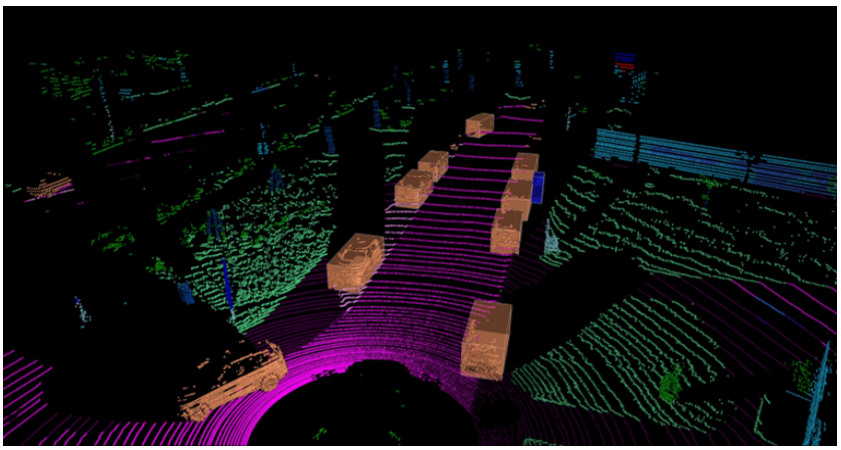

Figure 4.2: Example of the results of our 3D object detector.

Uncertainty Detection

The uncertainty module successfully predicts a spatial probability distribution for the edges of the bounding boxes surrounding tracked objects. Figure 4.5 displays the results of uncertainty detection on a parked vehicle. In this figure, the probability density distribution at the beginning of detection exhibits a wide spread, particularly around the vehicle's corners where no points are detected. As the vehicle approaches the parked car, the probability distribution gradually narrows. Once the optimal value for one

side of the bounding box's probability density function (pdf) is reached, it remains constant, ensuring an accurate prediction of the parked vehicle's size and position

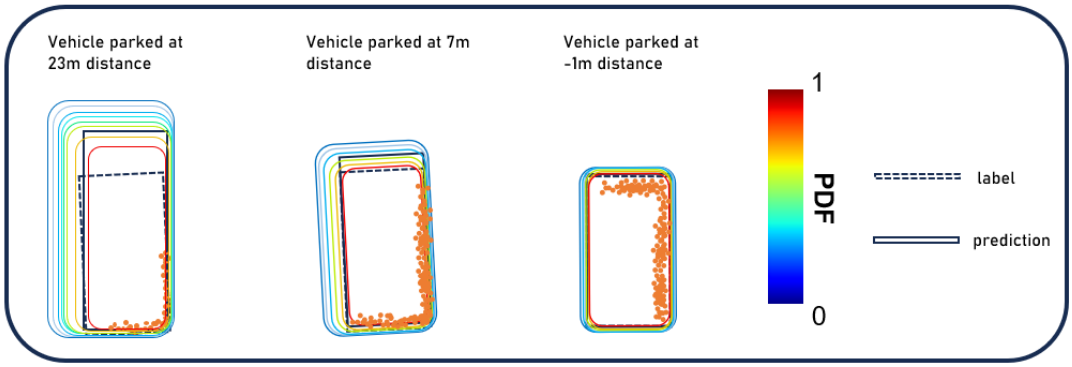

Figure 4.3: Results of the uncertainty detection module

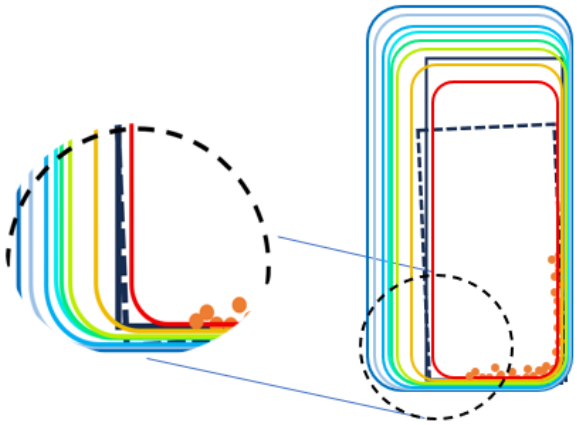

Figure 4.4: An example illustrating the spatial distribution of the bounding box for sides with fewer available points

The uncertainties always range from a minimum of -0.5m to a maximum of +0.5m from the bounding box edges. The figure below illustrates the distribution of the maximum size of the spatial distribution for all the detected bounding boxes in the KITTI dataset.

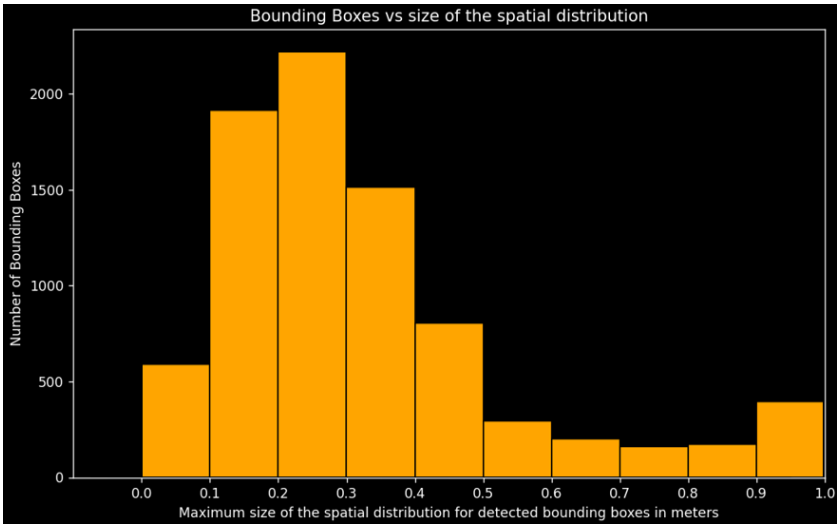

Figure 4.5: Statistical results of the maximum distribution

4.3. Qualitative analysis

Semantic Segmentation

The results obtained from the semantic segmentation module undeniably highlight the superiority of employing a multi-frame approach over a single-frame detector. The multi-frame detector excels in capturing temporal coherence between two frames, leading to significantly enhanced performance. However, it can be observed from the results that using more than two frames results in poorer performance for our model. This is primarily attributed to the movement of other vehicles, which introduces distortions during the joint voxelization process. Combining an image segmentation model with the 3D semantic segmentation model can potentially yield better results. Nevertheless, the current two-frame semantic segmentation model outperforms state-of-the-art 3D semantic segmentation modules.

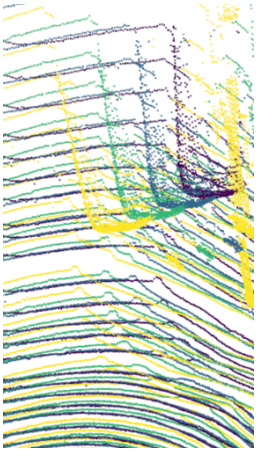

Figure 4.6: Distortion caused by the movement of the vehicle

Object detection

The object detection module outperforms state-of-the-art 3D object detectors when utilizing the 2-framed semantic segmentation network as its backbone. The performance of 3D object detection can be enhanced by injecting semantic score labels into the 3D object detection module, instead of predicting them with the 2-framed model. Additionally, incorporating the object detector into the LSTM model can further improve the overall model performance.

Spatial Uncertainty

The quantification of the spatial distribution for the bounding boxes constitutes a significant contribution to the field of automated driving. The uncertainty calculation model, which considers both epistemic and aleatoric uncertainties of the model, is correlated with the distribution of points in the point clouds. In cases where epistemic and aleatoric uncertainties were higher, the visualization of the spatial distributions reveals a wider spread, as can be shown in Figures 4.3 and 4.4. Figure 4.5 presents the statistical results of the maximum spatial distribution per bounding box. It is evident that the majority of bounding boxes converge towards a maximum spatial distribution ranging between 0.0 and 0.3 meters.

However, due to inaccuracies in the semantic segmentation module, the 3D object detection module, and the actual tracking of 3D objects, several bounding boxes showcase a spatial distribution between 0.9 and 1.0 meters. This is partly attributed to the movement of vehicles away from the ego-vehicle during data collection.

5

Conclusion

In this report, we have introduced an Uncertainty-Aware Object Detector (UA-LSTM) for object detection from LiDAR point clouds. The UA-LSTM model consists of three main modules: The feature extracting module, the object detecting module and the spatial uncertainty module. The LiDAR data first goes through the feature extracting module, which consists of a sparse Point-Voxel CNN and a LSTM network. In the feature extracting module, the point cloud is first fed through a single-frame Point-Voxel CNN model which gives us the semantic scores of the current LiDAR frame, where the semantic scores are a probabilistic type of output. Through a coarsely voxelization method, the semantic scores of the current frame and the semantic scores of the N previous frames gets coarsely voxelized. These coarsely voxelized inputs are fed into an adapted form of the Vanilla LSTM model where the fully connected layer is replaced by a lightweigth sparse Point-Voxel CNN, to improve the single frame semantic predictions. The devoxelized output from the LSTM is fed into the object detection module. The object detection model is an adapted form of PVCNN++ which takes the features (probabilsitic semantic scores) and the points to predict the position, sizes and orientation of the 3D objects. Finally, the spatial uncertainty module gives us the final representation of the bounding box and its corresponding uncertainties by .
In the experiments on the existing Kitti and Kitti-360 datasets, we have shown that the multi-frame object detectors, which considers the uncertainties of the model, outperforms the single frame uncertainties. We achieve state-of-the-art for sequential data in the Waymo open dataset. Also, we have introduces a network which not only predicts the uncertainties, but also gives an excellent representation of these uncertainties in the final output of the model.

Bibliography

[1] *Google scholar search*. [Online]. Available: https://scholar.google.com/scholar?q=3D+Object+detection+Object+OR+Detection+\%223D+Object+detection\%22&hl=nl&as_sdt=0\%2C5&inst=6173373803492361994&as_ylo=2020&as_yhi=.

[2] Y. Zhou and O. 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*, 2018, pp. 4490–4499.

[3] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[4] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.

[5] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[6] J. Mao *et al.*, "Voxel transformer for 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3164–3173.

[7] Z. Yang, Y. Zhou, Z. Chen, and J. Ngiam, "3d-man: 3d multi-frame attention network for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1863–1872.

[8] C. R. Qi *et al.*, "Offboard 3d object detection from point cloud sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6134–6144.

[9] R. Huang *et al.*, "An lstm approach to temporal 3d object detection in lidar point clouds," in *European Conference on Computer Vision*, Springer, 2020, pp. 266–282.

[10] Z. Luo, G. Zhang, C. Zhou, T. Liu, S. Lu, and L. Pan, "Transpillars: Coarse-to-fine aggregation for multi-frame 3d object detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023, pp. 4230–4239.

[11] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[12] J. Swiatkowski *et al.*, "The k-tied normal distribution: A compact parameterization of gaussian mean field posteriors in bayesian neural networks," in *International Conference on Machine Learning*, PMLR, 2020, pp. 9289–9299.

[13] 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)*, IEEE, 2018, pp. 3266–3273.

[14] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 677–12 686.

[15] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel cnn for efficient 3d deep learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *towards data science*, vol. 6, no. 12, pp. 310–316, 2017.

[17] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *18th international conference on pattern recognition (ICPR'06)*, IEEE, vol. 3, 2006, pp. 850–855.

[18] P. Ammirato and A. C. Berg, “A mask-rcnn baseline for probabilistic object detection,” *arXiv preprint arXiv:1908.03621*, 2019.

[19] C. Choy, J. Gwak, and S. Savarese, *4d spatio-temporal convnets: Minkowski convolutional neural networks*, 2019. arXiv: 1904.08755 [cs.CV].