Article

WCNN3D: Wavelet Convolutional Neural Network Based 3D Object Detection for Autonomous Driving

Simegnew Yihunie Alaba [1,*] and John E. Ball [2]

[1] Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS 39762, USA; sa1724@msstate.edu
[2] Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS 39762, USA; jeball@ece.msstate.edu
* Correspondence: sa1724@msstate.edu

Abstract: 3D object detection is crucial for autonomous driving to understand the driving environment. Since the pooling operation causes information loss in the standard CNN, we have designed a wavelet multiresolution analysis-based 3D object detection network without a pooling operation. Additionally, instead of using a single filter like the standard convolution, we use the lower-frequency and higher-frequency coefficients as a filter. These filters capture more relevant parts than a single filter, enlarging the receptive field. The model comprises a discrete wavelet transform (DWT) and an inverse wavelet transform (IWT) with skip connections to encourage feature reuse for contrasting and expanding layers. The IWT enriches the feature representation by fully recovering the lost details during the downsampling operation. Element-wise summation is used for the skip connections to decrease the computational burden. We train the model for the Haar and Daubechies (Db4) wavelets. The two-level wavelet decomposition result shows that we can build a lightweight model without losing significant performance. The experimental results on the KITTI's BEV and 3D evaluation benchmark show our model outperforms the Pointpillars base model by up to 14 % while reducing the number of trainable parameters. Code will be released.

Keywords: Autonomous Driving, Deep Learning, LIDAR Data, Wavelets, 3D Object Detection.

1. Introduction

LiDAR 3D object detection in robotics applications, such as autonomous vehicles (AVs), is essential for detecting other vehicles, pedestrians, and cyclists. Technology development companies, such as Waymo [1] and car manufacturing companies, such as Bosch [2], have tested vehicles equipped with different sensors, including LiDAR. Because of the rapid growth of deep learning (DL) methods in computer vision, research on 3D object detection has increased. Due to this growth, the performance of DL methods has improved. Among these, CNNs have shown a significant performance improvement and state-of-the-art performance in several computer vision tasks, such as image classification and 2D object detection. Although the pooling operation reduces the size of the features at every layer for strides more than one, it causes information loss, which can affect detection performance.

Another limitation of CNN is the lack of interpretability. A CNN is a universal function approximator. However, it is considered a black-box model because of nonlinear mapping and unclear working mechanisms [3]. This characteristic makes it difficult to understand how CNN learns and interacts with other models. Thus, it is hard to understand the soundness and weaknesses of the model and diagnose as well as correct potential problems [3]. Because of these problems, model development often relies on trial and error [4].

The discrete wavelet transform (DWT) can also enlarge the receptive field by considering more spatial context and employing multilevel decomposition filters (using low-frequency and high-frequency filters at each level). We can also enlarge the receptive field by using filters of larger sizes, but it increases the computational cost. Dilated convolution is implemented to enlarge the receptive field [5]. However, dilated convolution suffers from a gridding effect [6] because of the combination of pixels from different neighbors.

In the signal processing community, it is common to transform the spatial signal into a frequency domain representation, such as using a Fourier transform to use the dual property of convolution in the spatial domain and element-wise multiplication in the frequency domain. Rippel *et al.* [7] proposed a Fourier transform-based convolutional neural network to use the dual property and tested on the ImageNet [8] dataset. However, this is analogous to the conventional CNN in the frequency domain. The wavelet transform helps to keep both the spatial and frequency information. In this paper, we built a multiresolution analysis in the neural network with skip connections, WCNN3D: **W**avelet **C**onvolutional **N**eural **N**etwork Based **3D** Object Detection to mitigate the information loss due to the pooling operation. The standard 2D convolution operates in the spatial domain, which drops the high-frequency component. Designing wavelet-based CNN helps use high-frequency components, essential for learning and classifying images or other data. In a DWT, only the low-frequency component is decomposed into low-frequency and high-frequency components at each level of decomposition. But, we keep the high-frequency component in each multilevel DWT decomposition by concatenating the lower-frequency and higher-frequency components to enrich the features at each level of decomposition.

The point cloud data is encoded into point pillars to make the dataset suitable for 2D convolution and is an input to the backbone and detection modules. Pointpillars use pointNets to learn a representation of point clouds in a vertical column (pillars) [9] instead of using fixed encoders. We can apply 2D convolution to the encoded pointpillar data, which reduces the computation burden of 3D convolutions. The rest of the paper is structured as follows. Section 2 presents related work. The background information and intuition of wavelets are summarized in section 3. Section 4 summarizes the proposed model architecture and subnetworks, such as feature extraction and detection head, loss functions, and data augmentation techniques. Results and analyses of the experiment are presented in section 5. The last section 6 concludes the work.

2. Related Work

This section summarizes 3D object detection using camera images and LiDAR point clouds.

2.1. 3D Object Detection using Camera Images

Camera sensors are rich in texture and color information but lack depth information. So, 3D object detection using the camera is challenging. Different methods have been proposed to solve the lack of depth and minimize the performance drop. Some methods use Pseudo-LiDAR techniques, whereas others use stereo images and geometric constraints to solve the lack of depth in images [10]. The Pseudo-LiDAR methods predict the depth of each image pixel and generate Pseudo-LiDAR representation, such as [11,12]. The stereo image-based methods use stereo images to predict depth information and further generate 3D bounding box information, such as [13,14]. On the other hand, geometric-based methods use different geometric constraints such as ground planes and object shape [15,16] to estimate depth information.

Although different techniques have been proposed to solve the camera's lack of depth information, the performance of the proposed models is lower than 3D sensors such as LiDAR. For example, the Pseudo-LiDAR data does not perform similarly to LiDAR data because of the image-to-LiDAR generation error [10] .

2.2. 3D Object Detection Using LiDAR Point Clouds

Over the last few years, many LiDAR-based 3D object detection methods have been developed. The unstructured and sparse point cloud data is encoded into different representations to learn features: voxels, projections, and raw point cloud methods. VoxelNet [17] transforms point clouds into volumetric (voxel) representations and stacks the volumetric representations to form LiDAR point clouds. SECOND [18] improved voxelNet by introducing sparse convolution [19]. Voxel-FPN [20] also used voxel representation to do 3D object detection on point clouds. Pointpillars [9] encoded the point cloud data into pillar representation to solve the computational burden of voxel representation due to 3D convolution. Some works, such as PointNet [21] and PointNet++ [22] directly process the raw point cloud for 3D object detection. The raw point cloud implementation has a high computational burden because of the 3D convolutions, besides the sparse and unstructured nature of the point cloud. On the one hand, works such as BirdNet+ [23] projected the LiDAR point cloud into bird's-eye view (BEV) and applied 2D standard convolution on the projected data. Some methods use more than one sensor to solve the limitation of individual sensors and use the best out of each sensor. MV3D [24] and AVOD [25] fuse the projected BEV data with image data to enhance 3D object detection performance. Another technique is integrating more than one LiDAR data representation, such as voxel and raw point cloud. PV-RCNN [26] integrated the voxel representation and the raw point cloud to get the best of the two representations. Although the model showed a large margin performance improvement over the Pointpillars [9] network, the model's parameter is larger due to the 3D convolutions.

Many works adopt the pillar representation, such as [27–29] because of its ability to learn feature representation instead of extracting features using fixed encoders like VoxelNet [17]. Additionally, pillar representation is fast because we can apply 2D convolutions to the extracted features instead of 3D convolutions. Caine *et al.* [27] proposed a pseudo-labeling domain adaptation method for 3D object detection student-teacher network. Zhou *et al.* [29] presented a multiview fusion 3D object detection network from vertical column pillars and perspective view. In this work, besides using pillar encoding of point pillars to reduce the computational burden, we design a new wavelet-based convolutional neural network for 3D object detection to minimize the information loss because of the pooling operation in the standard 2D convolution and enlarge the receptive field.

Wavelet decomposition is well known and widely used in signal processing, but its usage in the computer vision community is limited. Works, such as [30], [31], [32] embed wavelets into the neural network to do 2D classification, image segmentation, and image restoration. Fujieda and Takayama [30] presented a multi-resolution analysis and CNNs combination for texture classification and image annotation. The authors achieved significant performance improvement over the CNN models with fewer parameters. Shen proposed WaveSNet [31] a wavelet integrated deep network for image segmentation applications. The author integrated discrete wavelet transform into U-Net [33], SegNet [34], and DeepLabv3+ [35] models for effective image segmentation. Liu *et al.* [32] proposed multi-level wavelet convolutional neural network (MWCNN) for image denoising, single image super-resolution, JPEG image artifacts removal, and classification. Even though the wavelet has shown significant performance in the above works and other signal processing tasks, it is not common in object detection, especially in 3D object detection. 3D object detection networks need to fulfill criteria such as being lightweight to run on real-time speed and accurate to give error-free information [36]. We introduce the first wavelet-based lightweight 3D object detection model for autonomous driving. This paper uses the term wavelet and discrete wavelet transforms interchangeably. We focus only on discrete wavelet transform in this work, so both terms refer to the discrete wavelet transform.

2.3. Contributions

We design a neural network by adding a high-frequency component to the standard spatial domain of CNN to enrich the feature representation. Additionally, we use the DWT as a downsampling operator to avoid information loss because of the pooling operation and enlarge the receptive field using the multiresolution filters. The proposed architecture is shown in Fig. 2. The main contributions of the paper can be summarized as follows.

1. To the best of our knowledge, our wavelet-based convolutional neural network model that incorporates the high-frequency component into the spatial domain to enhance performance is the first work in 3D object detection.
2. The model has high-frequency and low-frequency filters covering a large spatial domain to enlarge the receptive field, which improves the performance of the models.
3. We remove the standard pooling operation of CNN to avoid information loss and design wavelet-based CNN without pooling using the subsampling operation. The biorthogonal property of wavelets helps to perform subsampling without information loss. We apply IWT to enrich the feature maps for the detection head and fully recover the lost detail information.

We show the performance of the model on the KITTI dataset. The model significantly improves performance over the standard convolutional neural network models.

3. Wavelet based Convolutional Neural Network

Spectral approaches have been studied well in the signal and image processing community and achieved significant results, such as [37], [38], [31]. To use the advantage of wavelets, we design multiresolution [39] wavelet analysis based CNN for 3D object detection. The multiresolution analysis property of wavelets is advantageous in detecting objects of different scales.

The 2D wavelet decomposition on images has approximate (lower-frequency) and detailed (higher-frequency) coefficients. Then, at every stage of decomposition, the lower-frequency component convolves with the image and decomposes further into lower-frequency and higher-frequency coefficients. In a single-stage decomposition, there are four convolutional filters: f_{LL}, f_{LH}, f_{HL}, and f_{HH} as shown in Fig. 1. The high frequencies f_{LH}, f_{HL}, and f_{HH} represent the horizontal, vertical, and diagonal components of the decomposition, respectively. The subscripts L stands for low-pass filtered and H for high-pass filtered. The approximation information is represented by f_{LL}, which contains the maximum information of the decomposition. For example, convolving an image $\boldsymbol{x}$ with the four convolutional filters decomposes the image into subband images of x_{LL}, x_{LH}, x_{HL}, and x_{HH}. The convolutional filters differ based on the wavelet, such as a Haar or a Daubechies wavelet. For example, for the Haar wavelet, the filter coefficients are defined as:

$$f_{LL} = 1/2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, f_{LH} = 1/2 \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, f_{HL} = 1/2 \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, f_{HH} = 1/2 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},$$

The operation of DWT for subband images can be expressed as $x_{LL} = (f_{LL} * \boldsymbol{x})\downarrow 2$, $X_{LH} = (f_{LH} * \boldsymbol{x})\downarrow 2$, $x_{HL} = (f_{HL} * \boldsymbol{x})\downarrow 2$, and $x_{HH} = (f_{HH} * \boldsymbol{x})\downarrow 2$, where * is the convolution operator and $\downarrow 2$ is the downsampling/subsampling operator with a factor of two. Although there is a downsampling operation in the DWT, the original image/data $\boldsymbol{x}$ is fully recovered using the IWT without information loss due to the biorthogonal property of wavelets. In the standard convolution, deconvolution and maxunpooling operations are commonly used to increase the resolution of the feature map, but these operations cannot recover the lost data details. The IWT increases the resolution of the feature maps and helps recover the lost data details. Generally, we use the DWT as a downsampling operator and the IWT as an upsampling operator.

The multiresolution analysis of convolution followed by downsampling operation is equivalent to the standard convolution followed by pooling operation. For a convolution layer of a standard CNN, for a training set $\mathbf{x} = (x_0, x_1, ..., x_{n-1}) \in R^n$, we can get predicted labels $\mathbf{y} = (y_0, y_1, ..., y_{n-1}) \in R^n$.

$$y_n = \sum_{i \in N_n} w_i x_i + bias, \tag{1}$$

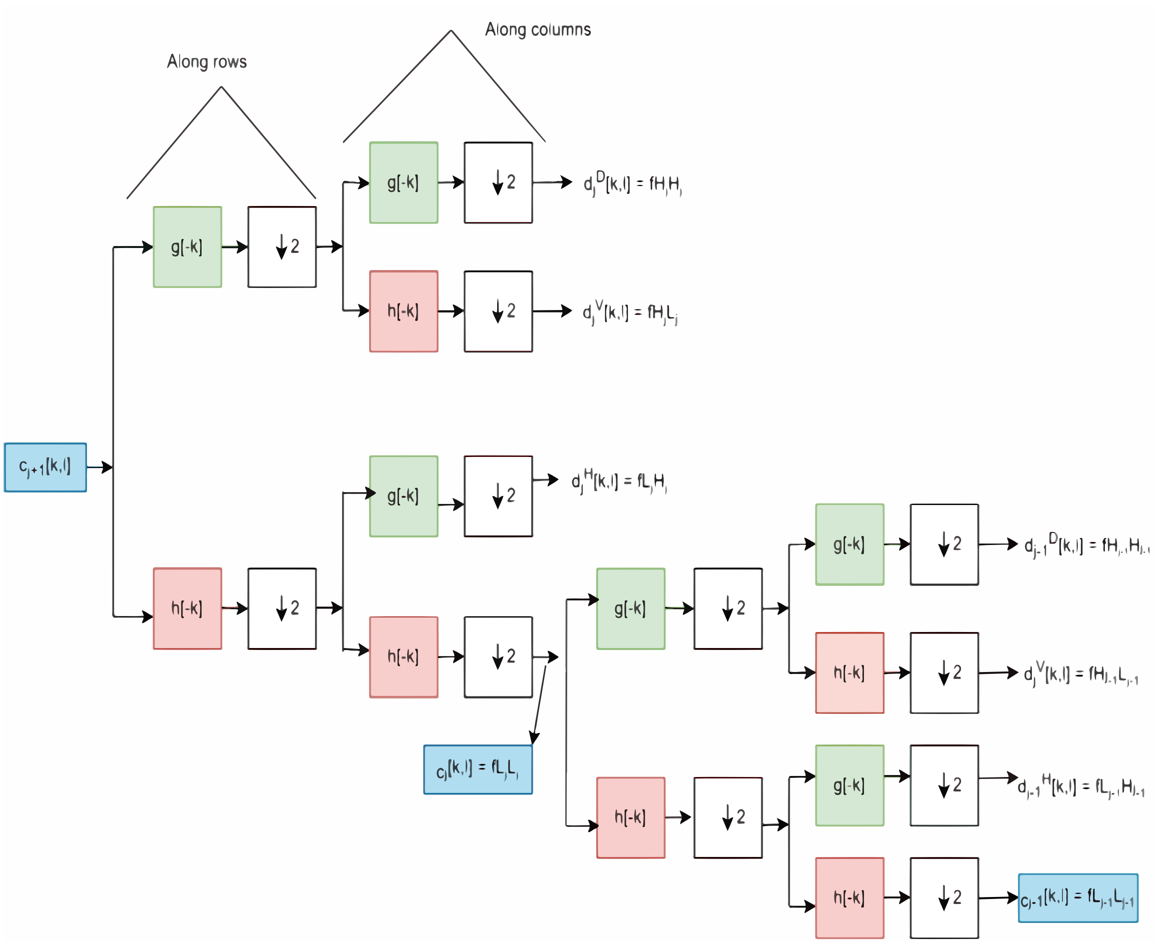

Figure 1. Two-stage 2D Wavelet Decomposition. The lower-frequency filter coefficient (approximation coefficient) decomposes into lower and higher filter coefficients at each stage of the wavelet decomposition. The decomposition starts with rows downsampling by a factor of two, followed by the same operation along with the columns (The operation can be done vice versa, starting with a column and then a row). Where c is the low-frequency coefficient (coarse coefficient), d is the high-frequency coefficient (detailed coefficient), g is the high-frequency filter, and h is the low-frequency filter. The $\downarrow 2$ operator is the downsampling/subsampling operator with a factor of two. At every stage of wavelet decomposition, the low-frequency coefficient decomposed into high-frequency and low-frequency coefficients.

where n is the number of samples, N_n is the set of indices of neighbors of x_i and w_i is the weight. The label y can be written using a cross-correlation (*i.e.*, convolution operator in most machine learning usage) * as :

$$y = x * w + bias, \tag{2}$$

where x and w are vectors.

The convolution operation of the input x with weight w is analogous to the DWT subband image x convolving with the filters. The subsampling layer (pooling layer) is applied immediately to the convolution layer to simplify the information in the standard CNN. For example, average pooling with a stride of two reduces the number of outputs by half to the corresponding number of inputs. We omit biases for the next operations to simplify the notation. We can express the generalized form of convolution followed by pooling operation as :

$$y = (x * w) \downarrow p, \tag{3}$$

where $\downarrow p$ is downsampling by a factor of p and $p > 1$. When $p = 2$, this operation is analogous to the multiresolution analysis of wavelets with lower-frequency and higher-frequency kernels. This operation can be written as :

$$\begin{aligned} x_l &= (x * f_l) \downarrow 2 \\ x_h &= (x * f_h) \downarrow 2, \end{aligned} \tag{4}$$

where f_l is the lower-frequency kernel (f_{LL}) and f_h is the high-frequency kernel, which consists of f_{LH}, f_{HL}, and f_{HH}. Equation (4) can be expressed as:

$$\begin{aligned} x_{LL} &= (x * f_{LL}) \downarrow 2 \\ x_{LH} &= (x * f_{LH}) \downarrow 2 \\ x_{HL} &= (x * f_{HL}) \downarrow 2 \\ x_{HH} &= (x * f_{HH}) \downarrow 2, \end{aligned} \tag{5}$$

The lower-frequency component is decomposed into hierarchical decomposition repeatedly for several decomposition levels. However, we use high-frequency and low-frequency kernels at each decomposition level to get richer information. At a decomposition level m, we can generalize the multiresolution analysis as:

$$\begin{aligned} x_l, m+1 &= (x * f_l, m) \downarrow 2 \\ x_h, m+1 &= (x * f_h, m) \downarrow 2, \end{aligned} \tag{6}$$

The standard CNN discards x_h, m and uses only the lower-frequency kernel.

$$x_l, m+1 = (x * f_l, m) \downarrow 2, \tag{7}$$

A single kernel is used in the standard CNN, but we used both low and high-frequency filters as a kernel. Using low and high-frequency filters increases the chance of taking into account more input pixels during the operation, which increases the field of view (receptive field). The receptive field is the size of a region in the input space that a particular feature looks at, which can be described by its central location and size. When the receptive field size increases, more relevant input parts are considered for detection [5]. The receptive field size of single path networks for each layer can be calculated using the following arithmetic [40]:

$$r_i = r_{i-1} + (f-1)j_{i-1}, \tag{8}$$

where r_i is the receptive field size for layer i, f is filter size of layer i, and j_{i-1} is the cumulative stride. The cumulative strides can be updated using $j_{out} = j_{in} * s$, where s is stride. To understand the intuition of how the DWT increases the receptive field, let's compare the Pointpillars [9] backbone and our backbone. All the filers are 3×3 in the pointpillars network; however, we used a combination of low and high-frequency filters. In the DWT, a low-frequency filter only is decomposed in each decomposition layer, but we use a combination of low and high-frequency filters to increase the receptive field size. Each low and high-frequency filter has a size of 2×2. When four of the filters, $f_{LL}, f_{LH}, f_{HL}, and\ f_{HH}$, are combined, we can get 8×8 filters, which is larger than pointpillars filter size.

Equation (8) works only for single-path networks (i.e., a single input to each layer). When multiple inputs to layer/layers exist, the receptive field size should be calculated for each input to the layer. Our network has multiple inputs from previous layers, both the downsampling and upsampling layers. The receptive field size increases when there are multiple inputs [41]. However, the detection /classification accuracy has a logarithmic relationship with the receptive field size [41]. Let's consider only filter size by keeping all other parameters, such as stride and padding, the same for both Pointpillars [9] and our network. From (8), we can see that the receptive field size of our network is greater than the Pointoillars network due to the larger filter size (i.e., 8×8 is greater than 3×3). However,

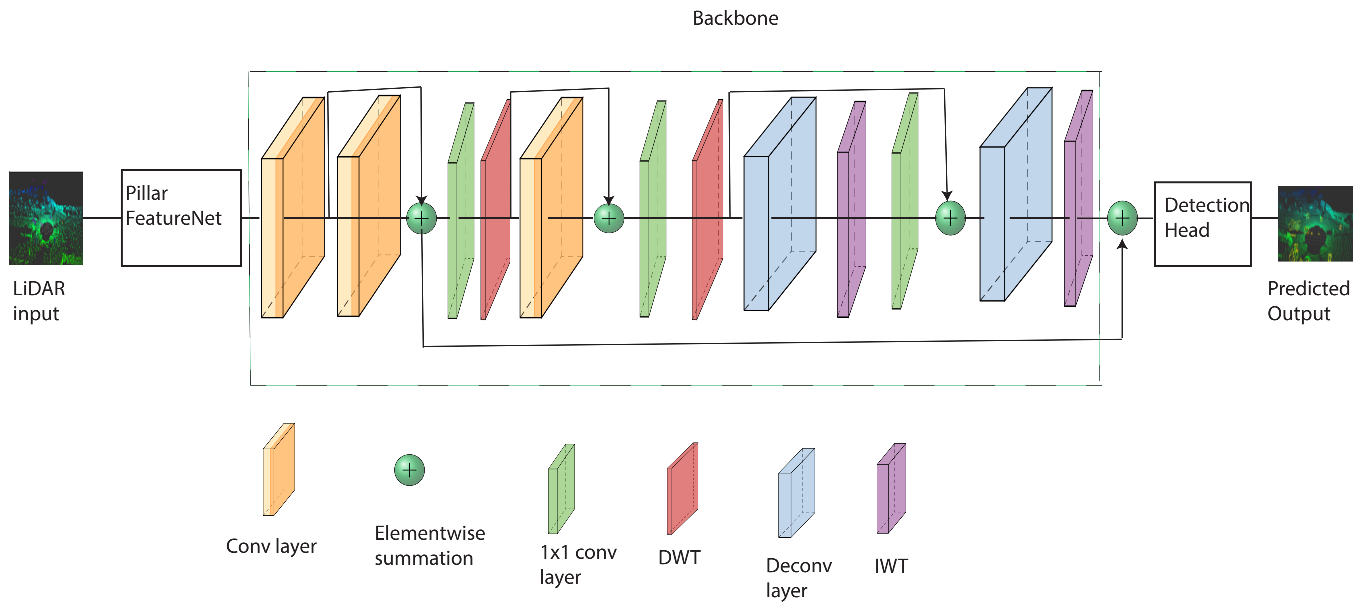

Figure 2. The architecture of the proposed WCNN3D object detection pedestrians and cyclists model for two levels of the wavelet decomposition. Each convolutional block comprises three convolutional layers at each level of decomposition. We also use 1x1 convolutions to reduce the channel dimension of feature maps. The raw LiDAR data is encoded through the Pillar Feature Network into a pillar representation. Then, the base network learns features using an expanding and contrasting wavelet convolutional network. The upsampling subnetwork recovers the lost details and increases the feature map of the downsampled features before detection. We do not apply the activation function for the last convolution layer. The upsampled features feed the detection head to predict 3D bounding boxes and classifications of objects. The convolution of down and up layers represent the convolutions of the contrasting and expanding layers, respectively.

due to the feature reuse design in the downsampling and upsampling layers, our proposed network has multiple inputs, potentially increasing the receptive field size further. Note that the receptive field is not the only contributing factor to the performance improvement of DL networks. Other factors, such as the number of layers, number of filters per layer, and batch normalization, contribute to the performance of a DL model. Therefore, using a set of kernels with lower-frequency and higher-frequency filters increases the receptive field. Integrating the higher frequency into the standard CNN outperforms the existing 3D object detection models, as shown in the experiment section 5.

4. WCNN3D Network Architecture and Implementation Details

The WCNN3D network comprises three major components: the feature network, the base network, and the detection head, as shown in Fig. 2. We use the pillar FeatureNet from Pointpillars [9] network to convert the raw point cloud into vertical columns' pseudo image representation. The base network, our main network, transforms the pillar features into high-level representation. Finally, the detection head predicts classes of objects and regresses 3D bounding boxes for objects.

4.1. Pillar FeatureNet

The point cloud data is converted into a pillar representation to avoid the computational burden of 3D convolutions. We use the pointpillar representation from Pointpillars [9] network because it reduces computational complexity by using 2D convolutions instead of 3D convolutions like VoxelNet [17] while maintaining high detection performance. The raw point cloud with x, y, z and reflectance is discretized into a set of pillars on an evenly spaced x-y grid. A pillar is a voxel that has unlimited spatial extent in the z-direction [9]. Then, the points in each pillar are decorated with reflectance (r), x_c, y_c, z_c, x_p, and y_p. The subscript c denotes the distance to the mean of all points in the pillar, whereas the subscript p denotes the offset from the pillar x and y centers. The LiDAR point at this stage is nine-dimensional (i.e., $x, y, z, r, x_c, y_c, z_c, x_p$, and y_p). Most of the pillars are empty due to the sparsity of LiDAR points, so a dense tensor of size (D, P, N) is created to reduce the sparsity of the decoded LiDAR point, where D denotes the dimension, P the number of nonempty pillars per sample, and N is the number of points per pillar. This operation reduces the memory complexity by focusing on the nonempty pillars only. Applying linear layer followed by BatchNorm [42] and ReLU [43], generates a tensor of size (C, P, N), where C denotes number of channels. The encoded feature scattered back to pseudo-image of size (C, H, W) where H and W denote the height and width, respectively. This representation avoids the computational burden because of 3D convolutions.

4.2. Base Network

We design a simple backbone network based on the DWT and IWT as shown in Fig. 2. The network comprises the downsampling and upsampling layers. The downsampling layer performs a 2D DWT, convolution, and ReLU activation. The layer also comprises skip-connections of features from previous layers. The upsampling network is designed using the IWT to increase the feature map resolution and fully recover lost details. The wavelet's lossless property helps fully recover the lost information during downsampling, which is essential for the detection head network. We also design skip-connections between consecutive layers of the downsampling and upsampling layers to enrich feature representation by encouraging feature reuse (knowledge preservation) and mitigating gradient problems for deep networks. An element-wise summation is applied for skip-connections instead of concatenation to decrease the computation burden. We also apply 1×1 convolution to reduce the channel dimensions and maintain the number of trainable parameters as small as possible while maintaining high detection performance. The batch normalization did not improve the performance, so we opted not to use it for the experiment. This condition might be due to the small number of batch sizes (two in this case because our machine cannot handle more than two batches). All weights are initialized randomly using the uniform distribution as in [44] and Pointpillars [9].

4.3. Detection Head

We use the Single Shot Detector (SSD) [45] as a detection head followed by regression for 3D bounding box generation. The SSD detection head comprises fully convolutional layers to predict four object predictions for each location (cell). Multiple predictions contain boundary box and confidence scores, essential to detect objects of different scales and aspect ratios. Due to these properties of the SSD, we choose to use the SSD detection head as our object detection head. Then, regression is done to get the 3D bounding box of objects. During the prediction level, non-maximum suppression (NMS) is also applied to suppress duplicate predictions.

4.4. Loss

The same loss function as of VoxelNet [17], SECOND [18], and Pointpillars [9] are used. The seven-point bounding box encoding technique (x, y, z, w, h, l, θ) is adopted. The x, y, and z are the center of coordinates; w, l, h are the width, length, and height, respectively and θ is the yaw rotation around the z-axis. The regression operation between ground truth and anchors can be defined as:

$$\Delta x = \frac{x^{gt}-x^a}{d^a},\ \Delta y = \frac{y^{gt}-y^a}{d^a}, \Delta z = \frac{z^{gt}-z^a}{d^a}$$
$$\Delta w = \log\frac{w^{gt}}{w^a},\ \Delta h = \log\frac{h^{gt}}{h^a},\ \Delta l = \log\frac{l^{gt}}{l^a}$$
$$\Delta\theta = \sin(w^{gt} - w^a), \text{ where the superscripts}$$

gt and a represent the ground truth and the anchor boxes, respectively. $d^a = \sqrt{(w^a)^2 + (l^a)^2}$ is the diagonal of the anchor box. The SmoothL1 loss and focal loss[46] are used as the localization loss and classification loss, respectively. We use a direction classifier loss, which uses the Softmax loss function, used for angle localization as in SECOND [18] and Pointpillars [9]. The SmoothL1 loss can be expressed as:

$$L_{loc} = \sum_{b \in (x,y,z,w,l,h,\theta)} SmoothL1(\Delta b),$$

where Δb can be Δx when b is x, Δy when b is y, $\Delta\theta$ when b is θ, etc.

Similarly, the focal loss can be expressed as follows.

$$L_{cls} = -\alpha_a (1 - p^a)^{\gamma} \log(p^a),$$

where γ is an anchor class probability and p^a is anchor class prediction. For fair comparison, we use the same values as [46] for $\alpha = 0.25$ and $\gamma = 2$. The total loss, which includes the localization loss, classification loss, and directional loss, can be expressed as:

$$L = \frac{1}{N_{pos}} (\beta_{loc} L_{loc} + \beta_{cls} L_{cls} + \beta_{dir} L_{dir}),$$

where N_{pos} is the number of positive anchors. We use $\beta_{loc} = 2$, $\beta_{cls} = 1$, and $\beta_{dir} = 0.2$ same as [46] and [9]. Adam optimization with a learning rate of $3 * 10^{-4}$ with a decay factor of 0.8 for every 15 epochs is used. We train the model for 160 and 320 epochs with a batch size of two. The model converges faster with fewer epochs too.

4.5. Dataset and Data Augmentation

4.5.1. Dataset

We use the KITTI dataset [47], which comprises 7,481 training and 7,518 testing samples. The model is trained with the LiDAR point clouds that comprise car, pedestrian, and cyclist categories. Each class is evaluated on three difficulty levels: easy, moderate, and hard, based on object size, occlusion, and truncation levels (truncation level refers to what portion of an object is outside the camera view). We divide the original training set into 3,712 training samples and 3,769 validation samples like [13] and Pointpillars [9]. Training the cars with one network and pedestrians and cyclists with another network has become common for 3D object detection [9,17,18,25]. Therefore, we follow the same practice to train cars for one network and pedestrians and cyclists for another network.

We use a maximum number of pillars of 12,000, a maximum number of points per pillar of 100, and a xy resolution of 0.16 m. We also train the model for a maximum of 15,000 pillars and a maximum number of points per pillar of 120. But, increasing the number of points adds a computational burden, so the above values are optimal for performance and computation. At inference time, NMS is applied to suppress detection of less than a given IOU value. For the KITTI dataset, cars are evaluated with an IOU of 0.7, whereas pedestrians and cyclists are evaluated with an IOU of 0.5. The x, y, and z range of [(0, 70.4), (-40, 40), (-3, 1)] meters, respectively are used for cars. The width of 1.6 m, length of 3.9 m, the height of 1.5 m, and a z-center of -1 m are used for the car's anchor box. Similarly, the x, y, z range for pedestrians and cyclists used are [(0, 48), (-20, 20), (-2.5, 0.5)] meters, respectively. The width of 0.6 m, length of 0.8 m, the height of 1.73 m, and a z-center of -0.6 m are used for the pedestrian anchor box, whereas width, length, and height of (0.6, 1.76, 1.73) meters, respectively with a z-center of -0.6 m are used for cyclist anchors. These values are commonly used on the KITTI dataset [48] for different works, such as [9,17,18].

4.5.2. 3D data augmentation

Data augmentation has been well studied in image classification problems than object detection models. An appropriate data augmentation, such as color transformation and geometric transformation, improves the generalization of a model. We followed SECOND [18] and Pointpillars [9] augmentation techniques. Pointpillars created a lookup table for all ground truth 3D bounding box classes. Then, randomly select 15 cars, zero pedestrians, and eight cyclists' ground truth samples for each sample. However, SECOND [18] used 8, 8, and 15 ground truth samples for pedestrians, cyclists, and cars, respectively. We use the Pintpillars [9] setting for this experiment because of its better performance compared to SECOND [18]. Then, each bounding box is rotated using a uniformly generated rotation angle in the range of [- $\pi/2$, $\pi/2$].

Table 1. 3D object detection result for Haar wavelet and Daubechies (Db4) on the KITTI test BEV detection benchmark. Note that '2L' refers to 2-level wavelet decomposition, '3L' refers to 3-level wavelet decomposition, and '4L' refers to 4-level wavelet decomposition. The best results are in boldface.

Methods	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [24]	-	86.62	78.93	69.80	-	-	-	-	-	-
F-pointNet [49]	65.20	91.17	84.67	74.77	57.13	49.57	45.48	77.26	61.37	53.78
PIXOR++ [50]	-	89.38	83.70	77.97	-	-	-	-	-	-
VoxelNet [17]	58.25	89.35	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
SECOND [18]	60.56	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
Pointpillars [9]	66.19	88.35	86.10	79.83	58.66	50.23	47.19	79.19	62.25	56.00
PV-RCNN [26]	70.04	**94.98**	**90.65**	86.14	59.86	50.57	46.74	82.49	68.89	62.41
WCNN3D (Haar-2L)(ours)	70.99	89.77	87.76	86.32	67.12	62.71	59.04	80.74	62.49	59.28
WCNN3D (Haar-3L)(ours)	71.02	90.05	87.51	86.11	69.20	63.29	59.26	81.54	62.26	58.32
WCNN3D (Haar-4L)(ours)	-	90.02	87.90	86.35	-	-	-	-	-	-
WCNN3D (Db4-2L)(ours)	71.66	90.11	87.83	86.27	**69.48**	**64.52**	**60.07**	**83.69**	62.63	59.45
WCNN3D (Db4-3L)(ours)	71.84	90.12	87.97	**86.46**	68.40	63.20	59.36	82.78	**64.34**	**60.29**
WCNN3D (Db4-4L)(ours)	-	90.20	88.04	86.31	-	-	-	-	-	

Table 2. 3D object detection result for Haar wavelet and Daubechies (Db4) wavelet on the KITTI test 3D detection benchmark. Note that '2L' refers to 2-level wavelet decomposition, '3L' refers to 3-level wavelet decomposition, and '4L' refers to 4-level wavelet decomposition. The best results are in boldface.

Methods	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [24]	-	74.97	63.63	54.0	-	-	-	-	-	-
F-pointNet [49]	56.04	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.17	49.01
VoxelNet [17]	49.05	77.47	65.11	57.73	39.48	33.69	31.5	61.22	48.36	44.37
SECOND [18]	56.69	83.13	73.66	66.20	41.07	42.56	37.29	70.51	53.85	46.90
Pointpillars [9]	59.20	79.05	74.99	68.30	52.08	43.53	41.49	75.78	59.07	52.92
PV-RCNN [26]	62.81	**90.25**	**81.43**	**76.82**	52.17	43.29	40.29	78.60	**63.71**	57.65
WCNN3D (Haar-2L)(ours)	64.36	87.09	77.39	75.49	58.62	54.57	50.02	79.56	61.13	57.37
WCNN3D (Haar-3L)(ours)	63.16	87.80	77.61	75.71	58.86	53.41	48.94	79.74	58.47	54.71
WCNN3D (Haar-4L)(ours)	-	87.84	77.67	76.00	-	-	-	-	-	-
WCNN3D (Db4-2L)(ours)	65.41	87.75	77.56	75.40	**61.93**	**57.67**	**52.06**	**82.74**	61.01	**57.66**
WCNN3D (Db4-3L)(ours)	64.16	88.57	78.04	76.16	57.88	53.75	49.82	80.14	60.70	56.70
WCNN3D (Db4-4L)(ours)	-	87.83	77.75	75.74	-	-	-	-	-	-

The local translation is also done with samples generated from a normal distribution of zero mean and standard deviation of [0.25, 0.25, 0.25] for x, y, and z. Finally, global augmentation is done to enrich the dataset further. Random mirror flip along the x-axis [9,51] is applied, and then a global rotation and scaling [9,17,18]. The global translation is applied that is drawn from a normal distribution with a mean of zero and standard deviation of [0.2, 0.2, 0.2] for x, y, and z. All experiments are done using NVIDIA GeForce RTX 2080 SUPER GPU running on Ubuntu 20.04.

5. Results and Discussions

We use the BEV and 3D KITTI datasets as official evaluation metrics for all experiments. Each class's detection is evaluated based on the three difficulty levels: easy, moderate, and hard. The KITTI dataset's mean average precision (mAP) is calculated based on the moderate difficulty level evaluation. Table 1 and Table 2 show the experimental results of our model compared with the base model Pointpillars [9] and other related state-of-the-art models. Our model outperforms other networks in mAP for both BEV and 3D KITTI dataset evaluation benchmarks. We train the car's network for two, three, and four levels of decomposition, but we do not train the pedestrians and cyclists' network for four levels of decomposition because of the small size of the objects to decompose after three levels of decomposition. The three difficult performance levels for cars, pedestrians, and cyclists show that our model's performance is much higher than other models for both Haar wavelet and Daubechies (Db4) wavelets, except for PV-RCNN for car networks. Although the cars' performance of the PV-RCNN model, which integrates the voxel form of LiDAR and the raw point cloud, is higher than our model, the pedestrians and cyclists' network performance is much lower than our model. Additionally, the mAP of our model is higher than PV-RCNN for both BEV and 3D KITTI evaluation metrics. For both Haar wavelet and Db4 wavelet, the four levels wavelets decomposition gives higher performance than the three levels and two levels wavelets decomposition, but the two levels of wavelet decomposition performance drop are insignificant compared to the decrease in training parameters. The Db4 performance is higher than the Haar wavelet performance, especially for pedestrians and cyclists.

Embedding the higher frequency into the spatial domain, which contains only the low-frequency, enriches the features and helps the detection head for classification and regression tasks. Additionally, the network's IWT part helps recover the lost details during downsampling. This recovery of lost details helps to increase the performance of the model. The performance improvement ratio of the model on smaller objects, such as pedestrians and cyclists, is greater than the cars' performance improvement, as shown in Table 1 and Table 2. Therefore, recovering the lost details using IWT is helpful, especially for small objects. Note that most DL models struggle to detect small objects. The moderate and hard difficulty levels of the KITTI dataset decrease the performance of all the models. However, the detection result of our model even increases at a higher rate for moderate and hard difficulty levels, which proves to add the higher frequency component helps the model generalize detection for such conditions. Even though PV-RCNN improves the performance of the Pointpillars by a large margin, the 3D convolution is still the bottleneck and is not a lightweight model for real-time processing. The major challenge of autonomous driving is increasing the performance while keeping the model lightweight. Our car model has 2.64 million, 3.33 million, and 5.6 million trainable parameters for two, three, and four decomposition levels. The pedestrians and cyclists model has two million and 2.97 million trainable parameters for two and three decomposition levels, respectively. The only lightweight model is the PointPillars network, with 5.9 million trainable parameters. The PV-RCNN model is not lightweight because of the 3D convolutions used for voxel processing. Therefore, our model is a lightweight model for even real-time processing. Our work will open the door to other researchers in the area to see the object detection network from the other direction to make 3D object detection interpretable, lightweight, and robust for real-time processing.

Our model produces an excellent result for cars, pedestrians, and cyclists, as shown in Fig. 3. The network can detect strongly overlapped and occluded objects but struggles with a few overlapped, occluded, and faraway objects. Generally, the performance of our network, especially for small objects such as pedestrians and cyclists, is promising.

6. Conclusions

In this work, we introduce WCNN3D, a new wavelet-based 3D object detection model for autonomous driving. The experimental results show that removing the pooling opera-

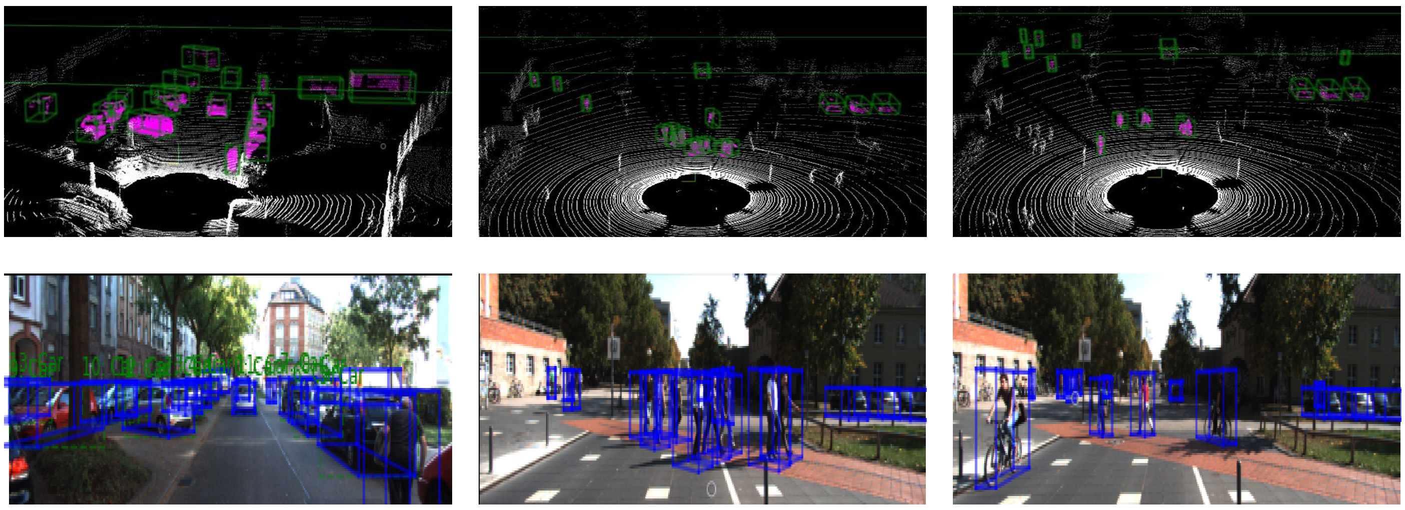

Figure 3. (a) cars, (b) pedestrians, (c) cyclists (from left to right). Qualitative Analysis of our LiDAR-based 3D object detection model on the KITTI dataset (zoomed view). For better visualization, we use the bird's eye view (top) and image perspective (bottom). The ground truth boxes are shown in blue, whereas the predicted boxes are in green. Even though the cars (a) are overlapped, and some are occluded, the network can detect them.

tion of CNN and replace with wavelet-based CNN increases the detection performance. Design wavelet-based CNN helps use the high-frequency component of the data that is throwaway by the standard convolution to enrich features and increase the detection performance of the model. We also use the lower-frequency and high-frequency coefficients of the wavelet as a kernel for convolution operation, which enlarges the receptive field. The detection result for two levels of wavelet decomposition shows that we can get lightweight models without losing much performance while decreasing the number of parameters. The model's overall performance on the BEV and 3D KITTI evaluation metrics shows that our model outperforms the pillar-based LiDAR 3D object detection models and gives competitive results to sensor fusion-based state-of-the-art models. Therefore, the advantage of wavelet-based CNN is multi-fold. Table 1 and Table 2 results show that our model is more suitable for small object detection than other models.

Author Contributions: Conceptualization, S.A.; methodology, S.A.; software, S.A.; validation, J.B.; formal analysis, S.A.; investigation, S.A.; data curation, S.A.; writing—original draft preparation, S.A.; writing—review and editing, J.B.; visualization, S.A.; supervision, J.B.; project administration, J.B.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

Funding: None

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: The KITTI public dataset is available here. Code will also be released.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
DL	Deep Learning
DWT	Discrete Wavelet Transform
IOU	Intersection Over Union
IWT	Inverse Wavelet Transform
mAP	Mean Average Precision
MWCNN	Multi-level Wavelet Convolutional Neural Network
NMS	Non-maximal Suppression
ReLU	Rectified Linear Unit
RCNN	Region-based Convolutional Neural Network
SSD	Single Shot MultiBox Detector

References

1. Waymo. Waymo driver-Waymo. Technical report, WAYMO, 2022.
2. Dechant, M. Self-driving car technology — Between man and machine. Technical report, Bosch, 2022.
3. Dong, Y.; Su, H.; Zhu, J.; Zhang, B. Improving interpretability of deep neural networks with semantic information. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4306–4314.
4. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In Proceedings of the European conference on computer vision. Springer, 2014, pp. 818–833.
5. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* **2015**.
6. Wang, P.; Chen, P.; Yuan, Y.; Liu, D.; Huang, Z.; Hou, X.; Cottrell, G. Understanding convolution for semantic segmentation. In Proceedings of the 2018 IEEE winter conference on applications of computer vision (WACV). IEEE, 2018, pp. 1451–1460.
7. Rippel, O.; Snoek, J.; Adams, R.P. Spectral Representations for Convolutional Neural Networks, 2015, [arXiv:stat.ML/1506.03767].
8. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *International journal of computer vision* **2015**, *115*, 211–252.
9. Lang, A.H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; Beijbom, O. Pointpillars: Fast encoders for object detection from point clouds. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12697–12705.
10. Alaba, S.; Ball, J. Deep Learning-based Image 3D Object Detection for Autonomous Driving **2022**.
11. Wang, Y.; Chao, W.L.; Garg, D.; Hariharan, B.; Campbell, M.; Weinberger, K.Q. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 8445–8453.
12. Ma, X.; Wang, Z.; Li, H.; Zhang, P.; Ouyang, W.; Fan, X. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6851–6860.
13. Chen, X.; Kundu, K.; Zhu, Y.; Berneshawi, A.G.; Ma, H.; Fidler, S.; Urtasun, R. 3d object proposals for accurate object class detection. In Proceedings of the Advances in Neural Information Processing Systems. Citeseer, 2015, pp. 424–432.
14. Shi, Y.; Mi, Z.; Guo, Y. Stereo CenterNet based 3D Object Detection for Autonomous Driving. *arXiv preprint arXiv:2103.11071* **2021**.
15. Mousavian, A.; Anguelov, D.; Flynn, J.; Kosecka, J. 3d bounding box estimation using deep learning and geometry. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7074–7082.
16. Liu, Z.; Wu, Z.; Tóth, R. Smoke: single-stage monocular 3d object detection via keypoint estimation. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 996–997.
17. Zhou, Y.; Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.
18. Yan, Y.; Mao, Y.; Li, B. Second: Sparsely embedded convolutional detection. *Sensors* **2018**, *18*, 3337.
19. Graham, B.; van der Maaten, L. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307* **2017**.
20. Kuang, H.; Wang, B.; An, J.; Zhang, M.; Zhang, Z. Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds. *Sensors* **2020**, *20*, 704.
21. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.
22. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413* **2017**.
23. Barrera, A.; Guindel, C.; Beltrán, J.; García, F. Birdnet+: End-to-end 3d object detection in lidar bird's eye view. In Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020, pp. 1–6.
24. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3d object detection network for autonomous driving. In Proceedings of the Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 1907–1915.

25. Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; Waslander, S.L. Joint 3d proposal generation and object detection from view aggregation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 1–8.
26. Shi, S.; Guo, C.; Jiang, L.; Wang, Z.; Shi, J.; Wang, X.; Li, H. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10529–10538.
27. Caine, B.; Roelofs, R.; Vasudevan, V.; Ngiam, J.; Chai, Y.; Chen, Z.; Shlens, J. Pseudo-labeling for Scalable 3D Object Detection. *arXiv preprint arXiv:2103.02093* **2021**.
28. Xu, Q.; Zhou, Y.; Wang, W.; Qi, C.R.; Anguelov, D. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 15446–15456.
29. Zhou, Y.; Sun, P.; Zhang, Y.; Anguelov, D.; Gao, J.; Ouyang, T.; Guo, J.; Ngiam, J.; Vasudevan, V. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In Proceedings of the Conference on Robot Learning. PMLR, 2020, pp. 923–932.
30. Fujieda, S.; Takayama, K.; Hachisuka, T. Wavelet convolutional neural networks. *arXiv preprint arXiv:1805.08620* **2018**.
31. Li, Q.; Shen, L. Wavesnet: Wavelet integrated deep networks for image segmentation. *arXiv preprint arXiv:2005.14461* **2020**.
32. Liu, P.; Zhang, H.; Lian, W.; Zuo, W. Multi-level wavelet convolutional neural networks. *IEEE Access* **2019**, *7*, 74973–74985.
33. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.
34. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* **2017**, *39*, 2481–2495.
35. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801–818.
36. Alaba, S.; Gurbuz, A.; Ball, J. A Comprehensive Survey of Deep Learning Multisensor Fusion-based 3D Object Detection for Autonomous Driving: Methods, Challenges, Open Issues, and Future Directions **2022**.
37. Fujieda, S.; Takayama, K.; Hachisuka, T. Wavelet convolutional neural networks for texture classification. *arXiv preprint arXiv:1707.07394* **2017**.
38. Kanchana, M.; Varalakshmi, P. Texture classification using discrete shearlet transform. *International Journal of Scientific Research* **2013**, *5*, 3.
39. Mallat, S.G. Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **1989**, *37*, 2091–2110.
40. Hien, D. A guide to receptive field arithmetic for convolutional neural networks. *medium. com* **2017**.
41. Araujo, A.; Norris, W.; Sim, J. Computing Receptive Fields of Convolutional Neural Networks. *Distill* **2019**. https://distill.pub/2019/compu receptive-fields, https://doi.org/10.23915/distill.00021.
42. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International conference on machine learning. PMLR, 2015, pp. 448–456.
43. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the Icml, 2010.
44. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
45. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European conference on computer vision. Springer, 2016, pp. 21–37.
46. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.
47. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In Proceedings of the 2012 IEEE conference on computer vision and pattern recognition. IEEE, 2012, pp. 3354–3361.
48. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
49. Qi, C.R.; Liu, W.; Wu, C.; Su, H.; Guibas, L.J. Frustum pointnets for 3d object detection from rgb-d data. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 918–927.
50. Yang, B.; Liang, M.; Urtasun, R. Hdnet: Exploiting hd maps for 3d object detection. In Proceedings of the Conference on Robot Learning. PMLR, 2018, pp. 146–155.
51. Yang, B.; Luo, W.; Urtasun, R. Pixor: Real-time 3d object detection from point clouds. In Proceedings of the Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2018, pp. 7652–7660.