Learning to Detect 3D Objects from Point Clouds in Real Time

Abhinav Sagar[*]
Vellore Institute of Technology
Vellore, Tamil Nadu, India
abhinavsagar4@gmail.com

Abstract

In this work, we address the problem of 3D object detection from point cloud data in real time. For autonomous vehicles to work, it is very important for the perception component to detect the real world objects with both high accuracy and fast inference. We propose a novel neural network architecture along with the training and optimization details for detecting 3D objects in point cloud data. We compare the results with different backbone architectures including the standard ones like VGG, ResNet, Inception with our backbone. Also we present the optimization and ablation studies including designing an efficient anchor. We use the Kitti 3D Bird's Eye View dataset for benchmarking and validating our results. Our work surpasses the state of the art in this domain both in terms of average precision and speed running at > 30 FPS. This makes it a feasible option to be deployed in real time applications including self driving cars.

1 Introduction

Machine learning has revolutionized the world and dramatically improved the lives of humans in various ways. Deep learning which is a branch of machine learning has been successfully applied to a lot of problems in the past decade including image classification, semantic segmentation, object detection, pose estimation, caption generation, image generation etc. A lot of work has been done in 2D object detection using convolutional neural networks. The object detection algorithms can be broadly grouped into the following two types:

1. Single stage detector - Yolo (Redmon et al., 2016), SSD (Liu et al., 2016).

2. Two stage detector - RCNN (Girshick et al., 2014), Fast RCNN (Girshick, 2015), Faster RCNN (Ren et al., 2015).

The difference between the two is that in the two stage detectors, the first stage uses region proposal networks to generate regions of interest and the second stage uses these regions of interest for object classification and bounding box regression. These are proven to have achieved better accuracy than the one stage architecture but comes at a tradeoff of more computational burden and time taken. On the other hand, a single stage detector uses the input image to directly learn the class wise probability and bounding box coordinates. Thus these architectures treat the object detection as a simple regression problem and thus are faster but less accurate.

There has also been a lot of work done on 3D object detection. Some of them use a camera based approach using either monocular or stereo images. Also work has been done by fusing the depth information on RGBD images taken from the camera. The main problem with camera based approach is the low accuracy achieved. Therefore lidar data has been proven to be a better alternative achieving

[*]Website of author - https://abhinavsagar.github.io/

higher accuracy and thus safety which is a primary concern for self driving cars. The challenge with using lidar data is that it produces data in the form of point clouds which have millions of points thus increasing the computational cost and processing time.

Point cloud data are of many types, of which the main type is 3D voxel grid. In this work we propose a single stage object detector using 2D Bird's Eye View (BEV) data. We used 2D Bird's Eye View in place of 3D voxel grid data because it is much less computationally heavy. This will also make our detector to be easily deployed to real work settings especially in case of self driving cars. Our detector accurately regresses the bounding box around objects in real time in birds eye view.

To validate our work, we benchmark our results on the publicly available 3D Kitti dataset (Geiger et al., 2012). For the evaluation metric, we use the class wise average precision. Our work beats the previous state of the art approaches for 3D object detection while also running at greater than 30 FPS. We also further show the learning and optimization aspects along with ablation study of this approach and present how it could potentially be generalized to other real world settings.

2 Related Work

2.1 Single-stage Object Detection

YOLO and SSD are the most popular approaches used in literature. YOLO (Redmon et al., 2016) divides the image into many rectangular grids and uses a sliding window which is a convolutional neural network to extract the features from the image. The window keeps on sliding and in this manner, it uses the context from other grids to determine if an object is present in the image. On the other hand SSD (Liu et al., 2016), uses anchors and multi scale convolutional feature maps instead of fully connected layers which are there in YOLO. SSD is more accurate and faster than YOLO.

Also a newer object detection architecture named RetinaNet (Lin et al., 2017b) shows that single stage architectures can outperform two stage architectures if the class imbalance problem is handled appropriately. RetinaNet uses focal loss to give more weight to the class which is present in lesser number and lesser weight to the class which is more frequently present. In this manner Retinanet combines the best of the two worlds i.e. good accuracy as well as faster run time.

2.2 Two-stage Object Detection

All of two stage object detection architectures use convolutional neural networks at its core to extract the features from the scene. Convolutional neural networks have been shown to work with remarkable accuracy on image classification problems. This is extended to object detection where a sliding window slides through an image thus extracting region proposals. A CNN runs through each of the region proposals to classify the object in each of the region proposals.

RCNN (Girshick et al., 2014) the first of the two stage architectures, uses a separate extraction and classification structures, Fast RCNN (Girshick, 2015) extracts the whole image feature map using pre trained CNN backbone and then predicts the confidence score and bounding box coordinates via ROI pooling operation. Faster RCNN (Ren et al., 2015) combines the above two approaches using both learning the feature map and generating the proposal in a single stage. This results in better accuracy and speed.

2.3 3D Object Detection from Point Clouds

Recently there have been a surge of papers on 3D object detection from various kinds of data like LIDAR, stereo etc. VOTE 3D (Qi et al., 2019) uses a sliding window on a 3D voxel grid to detect objects. The pre-trained model is fed to a SVM classifier later. VELOFCN (Li, 2017) projects 3D point cloud data to a perspective in the front view and gets a 2D depth map. The objects are detected by running a convolutional neural network on the depth map. MV3D (Qi et al., 2018) architecture also used a similar approach by combining the features extracted from multiple views like front view, birds eye view and camera view. These extracted features are passed through a CNN to detect 3D objects. PointNet (Qi et al., 2017) proposed an end-to-end classification neural network that directly takes a point cloud as input without any preprocessing and outputs class scores. (Zhou and Tuzel, 2018) subdivides the point cloud into 3D voxels and then transforms points within each voxel to a trainable feature vector that characterizes the shape information of the contained points. The representation vectors for each voxel stacks together and passes to a region proposal network to detect the objects.

In this work, our approach uses only the bird's eye view for 3D object detection in real time. The context of our work is in self driving cars but can be deployed in other settings as well.

We summarize our main contributions as follows:

• An approach to simultaneously detect and regress 3D bounding box over all the objects present in the image.

• A thorough analysis of backbone, optimization, anchors and loss function used in our architecture.

• Evaluation on the KITTI dataset shows we outperform all previous state-of-the-art methods in terms of average precision while running at >30 FPS.

3 Model

3.1 Dataset

For this work, we have used the Kitti dataset (Geiger et al., 2012) which contains LIDAR data taken from a sensor mounted in front of the car. Since the data contains millions of points and is of quite high resolution, processing is a challenge especially in real world situations. The task is to detect and regress a bounding box for 3D objects detected in real time. The dataset has 7481 training images and 7518 test point clouds comprising a total of labelled objects. The object detection performance is measured through average precision and IOU (Intersection over union) with threshold 0.7 for car class. The 3D object KITTI benchmark provides 3D bounding boxes for object classes including cars, vans, trucks, pedestrians and cyclists which are labelled manually in 3D point clouds on the basis of information from the camera.

KITTI also provides three detection evaluation levels: easy, moderate and hard, according to the object size, occlusion state and truncation level. The minimal pixel height for easy objects is 40px, which approximately corresponds to vehicles within 28m. For moderate and hard level objects are 25px, corresponding to a minimal distance of 47m.

3.2 Network Architecture

First we divided the point cloud data into 3D voxel grid cells. Our CNN backbone takes as input the image in the form of voxel and outputs a feature vector. The basic architecture consists of multiple residual blocks with skip connections connecting two adjacent blocks inspired by the famous ResNet architecture. In total there are 5 residual blocks. These blocks are in turn connected to upsampling blocks each except the first two residual blocks. The connections are inspired by the spatial pyramid pooling concept in which there is reduction in the number of channels as we go deeper. The last upsampling block is connected to two header network blocks. These blocks are further connected to two seperate separator blocks which in turn uses a bounding box regressor. Anchors are used in these header blocks to adjust the coordinates according to the size and shape of the body detected.

Each of the residual blocks are made up of: a fully connected layer followed by a non linearity activation function which is ReLu used in this case and a batch normalization layer. These layers are used for transforming each point in the voxel to a point wise feature vector. Element wise max-pooling layer is also used which extracts the maximum value from all the neighbouring pixel values when the filter is applied on the image. This operation is used for getting the locally aggregated features. Also a point wise concatenation operator is used which concatenates each point wise feature vector with the locally aggregated features.

In total our architecture has 14 convolutional layers and 3 max pooling layers. For our detector there are in total 7 parameters - three for the offset center coordinates, three for the offset dimensions and the last is for offset rotation angle. The network architecture is shown in Fig 1.

The detailed architecture of our model layerwise with filters, size, stride, padding, extension, input and output shapes is shown in Table 1.

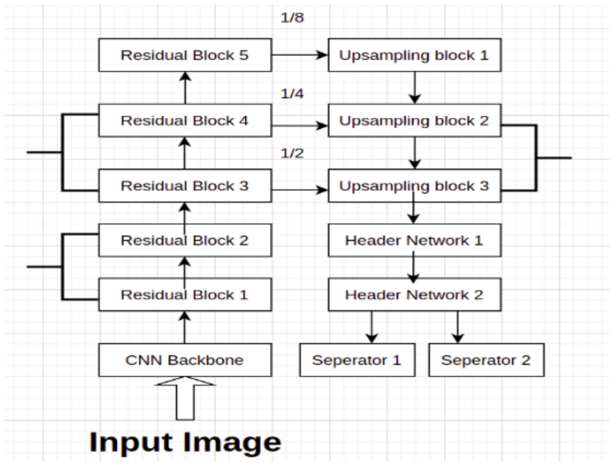


Figure 1: Our network architecture

Table 1: Detailed architecture layerwise of the neural network that is used for the training

Layer	Filter	Size	Stride/Padding	Extension
Convolution1	3×3	64	1/1	ReLU, BN
Convolution2	3×3	128	1/1	ReLU, BN
MaxPooling1	2×2	-	1/0	-
Convolution3	3×3	128	1/1	ReLU, BN
Convolution4	2×2	64	1/1	ReLU, BN
Convolution5	2×2	128	1/1	ReLU, BN
MaxPooling2	3×3	-	1/0	-
Convolution6	3×3	256	1/1	ReLU, BN
Convolution7	2×2	128	1/1	ReLU, BN
Convolution8	2×2	256	1/1	ReLU, BN
MaxPooling3	2×2	-	1/0	-
Convolution9	1×1	512	1/1	ReLU, BN
Convolution10	2×2	256	1/1	ReLU, BN
Convolution11	2×2	512	1/0	ReLU, BN
Convolution12	2×2	256	1/0	-

3.3 Backbone Network

Multiple blocks of 2D convolutional layers are used to generate feature maps at different resolutions. Also transposed convolutional layers are used to get the feature maps with the same dimension as that of the output layer. At each anchor position, 1×1 convolutions are used which are concatenated. The output can be decomposed into 2 parts - regression map and probability score map. The receptive field of the probability score map is kept higher to capture the various sizes of objects present in the image. The compared results with different backbones is shown in Table 2.

Table 2: Ablation study of different backbone networks

Background Network	Average Precision
VGG16	54.46
VGG19	53.68
ResNet50	53.03
InceptionV3	54.25
DenseNet169	53.69
Ours	55.36

4 Experiments

4.1 Anchors

Anchors are very important for efficient object detection. These are basically prior beliefs containing information of the size for the detected object, its position is the image, its pose, its orientation etc. Anchors of multiple shape, size are more stable, also helps in reducing the computational burden and time taken by the model. We have chosen two anchors for each of the classes as shown in Table 3, Table 4 and Table 5:

Table 3: Car anchors

Height(m)	Width(m)	Length(m)	Rotation(Theta)
1.6	1.6	4	0
1.6	1.6	1.6	90

Table 4: Pedestrian anchors

Height(m)	Width(m)	Length(m)	Rotation(Theta)
1.7	0.5	0.7	0
1.7	1.5	0.7	90

Table 5: Cyclist anchors

Height(m)	Width(m)	Length(m)	Rotation(Theta)
1.6	0.7	2	0
1.6	0.7	2	90

4.2 Loss

A vector $s = (x, y, z, l, h, w, \theta)$ represents 3D bounding box center coordinates, height, width, length and yaw respectively. The geometric relations between various parameters is illustrated in the equation below where s represents the ground truth vector and a represents the anchor vector. The localization regression between ground truth and anchors are defined in Equation 1:

$$\begin{array}{ll} \Delta x = \frac{x_s - x_a}{\sqrt{l^2+w^2}} & \Delta z_b = z_s - \frac{h_s}{2} - z_a + \frac{h_a}{2} \\ \Delta y = \frac{y_s - y_a}{\sqrt{l^2+w^2}} & \Delta z_t = z_s + \frac{h_s}{2} - z_a - \frac{h_a}{2} \\ \Delta l = \log \frac{l_s}{l_a} & \Delta w = \log \frac{w_s}{w_a} \\ \Delta \zeta = \left|\sin\left(\theta_s - \theta_a\right)\right| & \Delta \eta = \cos\left(\theta_s - \theta_a\right) \end{array} \tag{1}$$

Since the angle localization loss cannot distinguish the bounding boxes which are flipped, we use a softmax classification loss as shown for both positive and negative anchors. For the object classification, we have used focal loss as shown in Equation 2 and Equation 3 respectively:

$$\mathcal{L}_{pos} = -\alpha_a \left(1 - p^a\right)^\gamma \log p^a \tag{2}$$

$$\mathcal{L}_{neg} = -\alpha_a \left(1 - p^a\right)^\gamma \log p^a \tag{3}$$

We used Intersection Over Union (IOU) for evaluating the performance of our network. All the positive anchors have an IOU value above 0.60 while those with less than 0.45 are treated as negative anchors. We used binary cross entropy loss for detection and a variant of huber loss for regression.

Let i and j denote the positive and negative anchors and let p denote the sigmoid activation for the classification network. Let pos represent the positive regression anchors and neg the negative regression anchors. The overall loss function is shown in Equation 4:

$$L = \alpha \frac{1}{N} \sum_i L_c\left(p_i^{pos}, 1\right) + \beta \frac{1}{N} \sum_j L_c\left(p_j^{neg}, 0\right) + \gamma \frac{1}{N} \sum_k \left(L_r\left(l, l^*\right) + L\left(h, h^*\right) + L_c\left(w, w^*\right)\right) \tag{4}$$

Here α, β and γ are the hyperparameters which we tuned using bayesian optimization. The optimal value of α, β and γ comes out to be 0.36, 0.14 and 0.63 respectively.

5 Results

Table 6 compares the results of the LIDAR based 3D object detector on KITTI testing set. For the fair comparison VeloFCN, 3DFCN, MV3D architectures have been shown with their inference time, class wise average precision metric for all the three categories including easy, medium and hard.

Table 6: Evaluation results of LIDAR based 3D object detectors on KITTI BEV Object Detection testing set

Method	Time(ms)	AP easy(%)	AP moderate(%)	AP hard(%)
VeloFCN (Li et al., 2016)	1000	0.15	0.33	0.47
3DFCN (Li, 2017)	5000	69.94	62.54	55.94
MV3D (Chen et al., 2017)	240	85.82	77.00	68.94
Ours	30	84.38	78.27	71.30

Table 7. compares our result with F-PointNet, AVOD and VoxelNet architectures. FPS along with class wise average precision is used for all three cases i.e. easy, medium and hard. All the three classes here have been compared against i.e. car, pedestrian and cyclist.

Table 7: Performance comparison for 3D object detection. APs (in %) for our experimental setup compared to current leading methods.

Method	FPS	Easy	Med	Hard	Easy	Med	Hard	Easy	Med	Hard
F-PointNet (Qi et al., 2018)	5.9	81.20	70.39	62.19	51.21	44.89	40.23	71.96	56.77	50.39
AVOD (Ku et al., 2018)	12.5	73.59	65.78	58.38	38.28	31.51	26.98	60.11	44.90	38.80
VoxelNet (Zhou and Tuzel, 2018)	4.3	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
Ours	30	77.72	74.00	63.01	41.79	45.70	42.92	68.17	58.32	54.30

The performance comparison of 3D object detection on the KITTI 3D object detection benchmark and BEV benchmark for the class car is shown in Table 8.

Table 8: Performance comparison of 3D object detection on the KITTI 3D object detection benchmark and BEV benchmark for the class car. Values are average precision (AP) scores on the test set.

Method	Modality	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D (Chen et al., 2017)	RGB + LiDAR	71.09	62.35	55.12	86.02	76.90	68.49
Voxelnet (Zhou and Tuzel, 2018)	LiDAR	77.47	65.11	57.73	89.35	79.26	77.39
F-PointNet (Qi et al., 2018)	RGB + LiDAR	81.20	70.39	62.19	88.70	84.00	75.33
AVOD (Ku et al., 2018)	RGB + LiDAR	81.94	71.88	66.38	88.53	83.79	77.90
PointRCNN (Shi et al., 2019)	LiDAR	85.94	75.76	68.32	89.47	85.68	79.10
MMF (Liang et al., 2019)	RGB + LiDAR	86.81	76.75	68.41	89.49	87.47	79.10
Ours	LiDAR	87.27	77.14	68.86	90.20	86.67	79.25

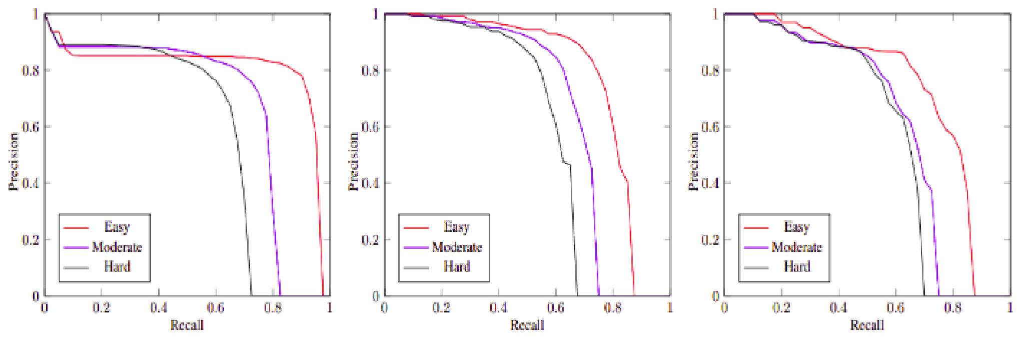
Our results are considerably better than the previous state of the art approaches.

5.1 Precision and Recall

Both precision and recall are very important metrics that evaluate the efficiency of the object detector. The precision measures how many of the predicted objects were classified correctly. A precision score of 1.0 for a class indicates that every object predicted, is classified correctly in that class. While recall is the fraction of correct predictions among the all observations in the actual class. A recall score of 1.0 for a class indicates every object in that class has been found and predicted correctly by the model.

5.2 Average Precision

The ideal value of precision and recall is 1. Since it is not possible to get perfect values, the closer the metrics ie precision and recall is to 1, the better our model is performing, It's often seen that there is a tradeoff between precision and recall ie if we are optimizing for precision, recall value gets less and if we are trying to improve recall, precision value becomes less. So our task is to balance both and note that threshold point. Average precision is the average value of precision for the sampled points at various recall threshold values. The precision - recall curve for 3D object detection for the 3 classes i.e. cars, pedestrians and cyclists for all the three categories i.e. easy, moderate and hard are shown in Fig 2. The closer the curve is to (1,1), the higher performance of the model is.



(a) Cars (b) Pedestrians (c) Cyclists

Figure 2: Precision-recall curve for 3D detection in a) Cars b) Pedestrian c) Cyclists

A sample of the predicted 3D detection from the KITTI validation dataset is shown in Figure 3:

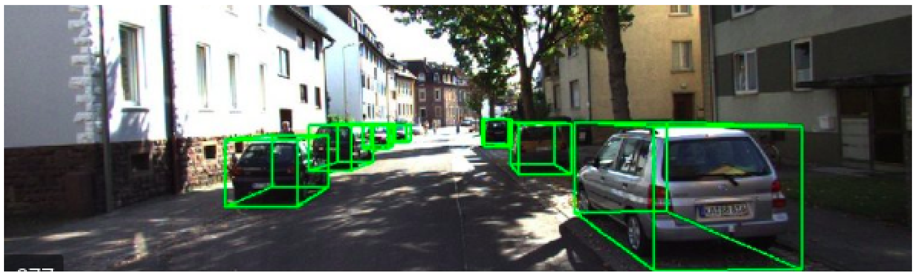


Figure 3: 3D detection from the KITTI validation dataset projected onto an image

Finally we present the results for 3D object detection results on KITTI validation set in Fig 4. The ground truth bounding boxes are shown in blue and the predicted bounding boxes are shown in orange.

Note that our model is based only on LiDAR data. For better visualization the 3D bounding boxes are projected on to the bird's eye view and the images.

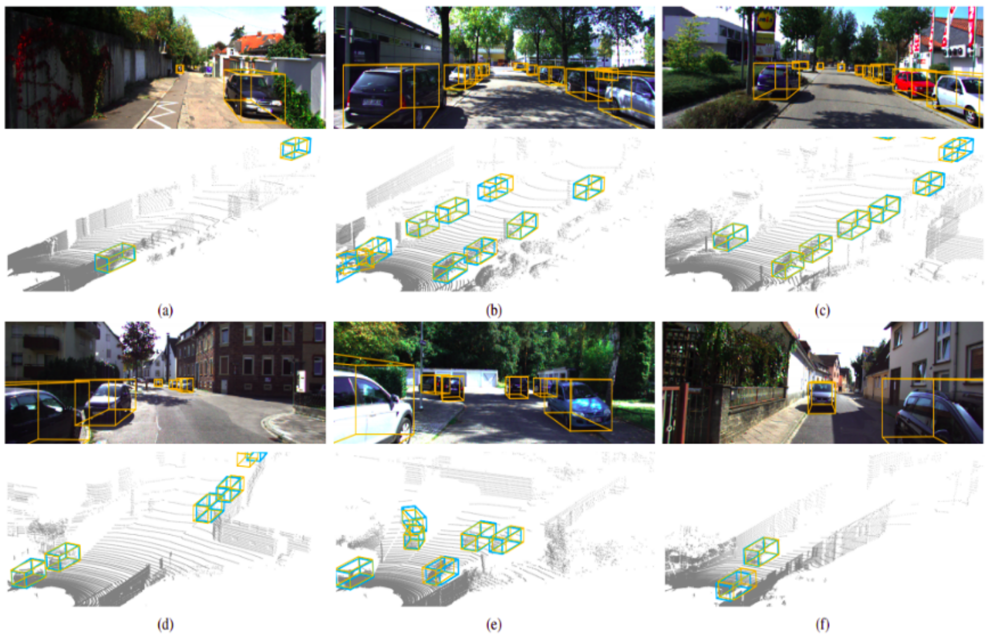


Figure 4: Results. Predicted 3D bounding boxes are drawn in orange, while ground truths are in blue

6 Conclusions

In this paper, we proposed a 3D object detection architecture which works in real time using LIDAR point cloud data. For making efficient computation, our architecture uses a single stage type neural network with bird's view representation. We presented the architecture details as well as optimization done. We evaluate our study on the KITTI benchmark dataset and show that our work outperforms previous state of the art approaches. As for the evaluation metric, we chose class wise average precision. The model runs at faster than 30 FPS and hence can be used in autonomous driving applications where safety is a major challenge.

Acknowledgments

We would like to thank Nvidia for providing the GPUs.

References

X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

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

X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9): 1904–1916, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

S. Hong, B. Roh, K.-H. Kim, Y. Cheon, and M. Park. Pvanet: Lightweight deep neural networks for real-time object detection. *arXiv preprint arXiv:1611.08588*, 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

B. Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017.

B. Li, T. Zhang, and T. Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017a.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017b.

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9277–9286, 2019.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607, 2015.

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*, pages 7652–7660, 2018.

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*, pages 4490–4499, 2018.