YOLO4D: A Spatio-temporal Approach for Real-time Multi-object Detection and Classification from LiDAR Point Clouds

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, YOLO4D is presented for Spatio-temporal Real-time 3D Multi-object detection and classification from LiDAR point clouds. Automated Driving dynamic scenarios are rich in temporal information. Most of the current 3D Object Detection approaches are focused on processing the spatial sensory features, either in 2D or 3D spaces, while the temporal factor is not fully exploited yet, especially from 3D LiDAR point clouds. In YOLO4D approach, the 3D LiDAR point clouds are aggregated over time as a 4D tensor; 3D space dimensions in addition to the time dimension, which is fed to a one-shot fully convolutional detector, based on YOLO v2. The outputs are the oriented 3D Object Bounding Box information, together with the object class. Two different techniques are evaluated to incorporate the temporal dimension; recurrence and frame stacking. The experiments conducted on KITTI dataset, show the advantages of incorporating the temporal dimension.

1 Introduction

Environmental modeling and perception is a critical component to automated driving pipeline. In order to have efficient planning in complex scenarios, 3D object detection is essential, to get information about the objects extent and range in the 3D space. Deep Learning is becoming the trend for real-time object detection and classification [16][18] [1] [17][10]. LiDAR sensors are used to perceive the 3D nature of objects, where the sensor provides a 3D point cloud (PCL) representing the range of reflected laser beams of the surrounding objects. The task of 3D object detection and classification from such a point cloud is challenging. On one hand, the point cloud is sparse, since not all beams are reflected, and noisy due to imperfect reflections and echoes. On the other hand, the 3D point cloud does not have the color and texture features that characterize the object classes as in the case of 2D camera perspective images. Such complexity, in addition to the dynamic nature of the environment, motivates our work to incorporate the temporal factor in addition to the spatial features of the input 3D LiDAR point clouds. In this paper, we present YOLO4D; a Spatio-temporal extension of the work done in YOLO3D[1] for real-time multi-object detection and classification from 3D LIDAR point clouds, were YOLO3D is extended with Convolutional LSTM [20] for temporal features aggregating. The 3D LiDAR point clouds are aggregated over time as a 4D tensor; 3D space dimensions in addition to the time dimension. The output is the oriented 3D object bounding Box (OBB) coordinates of the center, in addition to its length (L), width (W), height (H) and orientation (yaw), together with the objects classes and confidence scores. The evaluation of the Spatio-temporal approach is performed on KITTI dataset [4]. We compare two approaches to incorporate the temporal factor; YOLO4D and frame stacking. The experimental results show the advantage of adding the temporal aggregation in addition to spatial features.

The rest of the paper is organized as follows; first, we discuss the related work, followed by the Spatio-

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

temporal approach, with the proposed network architecture. Finally, we present the experimental results and evaluation of different techniques on KITTI dataset.

2 Related Work

3D Object Detection: Recent works in 3D object detection depends on 3D sensors like LiDAR to take advantage of accurate depth information. Different data representations are introduced, like projecting point cloud in 2D view (Bird Eye View, Front View) such as PIXOR [21] , [8] and MV3D [3]. PIXOR assumes that all objects lie on the same ground but in this work, we do not make this assumption. We regress the height information freely. Some [2] convert the point cloud to a front view depth map. Others like [7] and [22], convert the point cloud to voxels which produce a highly sparse representation leading to inefficient object detection. Finally, some work inspired by PointNet [14] and PointNet++[15] process the LiDAR PCL as an unordered set like PointCNN [9]. These methods suffer from heavy computations. All these methods do not take advantage of the temporal information to produce high-quality 3D bounding boxes. In this work we extend the work done in YOLO3D [1] using the same input representation for the LiDAR PCL but with the temporal information.

Object detection based spatial temporal recurrence: While some papers addressed the use of recurrence as Spatial-temporal feature extractor alongside with the object detection [19] [13]. They focused on improving visual tracking in videos. For example, in ROLO [13] they used the same architecture of YOLO v1 [16] with an LSTM [5] layer added at the end. The network consumes as input raw videos and returns 2D tracked bounding box. In contrast, our proposed architecture, that takes a sequence of 3D LiDAR point cloud processed as introduced in [1] alongside with the temporal information from the previous frames by utilizing Convolutional LSTM [20] as a Spatio-temporal fusion layer to produce 3D bounding boxes. In [19] they divided the problem into two stages. In the first stage, they performed the detection using YOLO v1 [16]. In the second stage, they used a combination of fully connected layers and GRUs. In our work, we utilize YOLO v2 [18] with Convolutional LSTM as a single fully convolutional neural network. Recently, Fast and Furious [11] tried to incubate the time with 3D voxels using multi-task learning. In our work, we are using Convolutional LSTM instead of 3D convolutions with much less input processing complexity by using a single channeled bird eye view as an input to produce the 3D information which gained us a faster inference time of 20ms.

3 Spatio-temporal 3D object detection approach

In this section, the approach for Spatio-temporal 3D object detection is described. The main intuition behind our work is to leverage not only the spatial but also the temporal information in input sequences for more accurate object detection. For encoding temporal sequences, we adopted two different approaches, YOLO4D and frame stacking. Both approaches encode the temporal information in different ways. Frame stacking is a simple method that depends on the input layer, while, YOLO4D employs Convolutional LSTM.

3.1 Spatial 3D object detection

Following the work done in YOLO3D, the point cloud is projected into a bird's eye view (BEV) grid map. The orientation of the bounding boxes is normalized and used as a single regressed value. For 3D bounding box regressions, two regression terms are added to the original YOLO architecture, the z coordinate of the center, and the height h of the box. The average 3D box dimensions for each object class is calculated on the training dataset and used as anchors. The loss for the 3D oriented boxes is an extension to the original YOLO loss for 2D boxes. The total loss is formulated as the weighted summation of the mean squared error over the 3D coordinates, dimensions, the mean squared error over the angle, the confidence score, and the cross-entropy loss over the object classes.

3.2 Temporal aggregation

The dataset $\mathcal{T}$ is composed of number of k scenarios $\{S^0, S^1, ...S^{k-1}\}$, a scenario S^i consists of a sequence of n frames $\{I^i_0, I^i_1, ...I^i_{n-1}\}$, with the corresponding list of 3D bounding boxes targets for

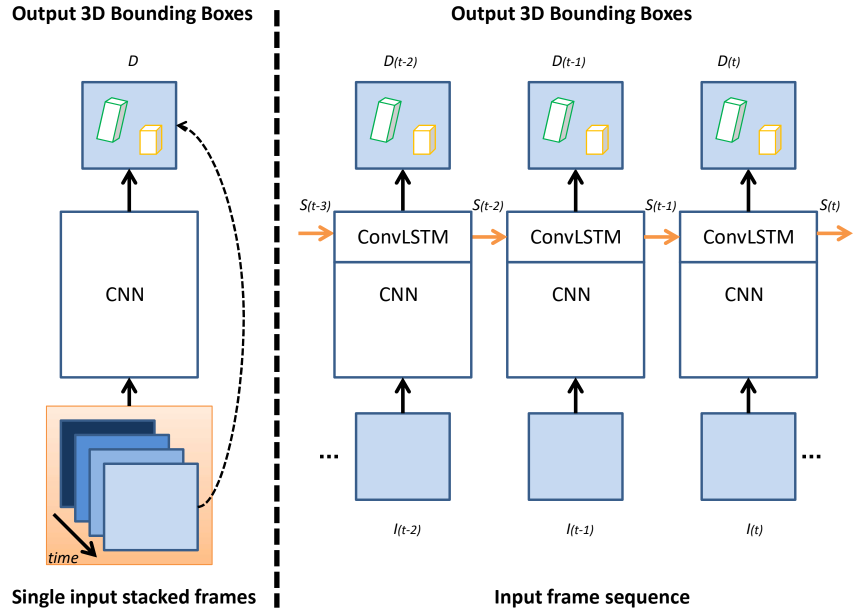

Figure 1: **Left**: Frame stacking architecture; **Right**: Convolutional LSTM architecture.

each frame, $\{D_0^i, D_1^i, ...D_{n-1}^i\}$. Each scenario is divided into a number of short clips of m frames. The frames of the clip j from scenario i is defined as $Q_j^i = \{I_{t_j}^i, I_{t_j+1}^i, ...I_{t_j+m-1}^i\}$.

3.2.1 Frame stacking

In frame stacking, frames of each clip are stacked in-order together and presented as a single input to the object detection network for training. Accordingly, the frame stacking approach encodes the temporal information indirectly by reshaping the input by increasing its depth to represent the changes over time. During the training process, it is up to the network to learn the temporal information from the input stacked frames without encoding hidden state through recurrent layers. The loss is similar to the YOLO3D architecture for a single frame input, since the predicted 3D bounding boxes depend only on a single stacked input.

3.2.2 Convolutional LSTM

In YOLO4D architecture, a Convolutional LSTM layer is injected directly into YOLO3D single frame architecture, see section 4.3 for more details. Convolutional LSTM allows the network to learn both spatial and temporal information. The network is trained on the same sequences used for frame stacking approach, leading to a model that is capable of detecting objects in temporal streams of input frames. The prediction model can be considered as a function F, parameterized by θ, that maps an input frame and the previous state to a list of 3D bounding boxes as shown in equation 1.

$$\mathcal{F}_\theta(I_t, s_{t-1}) = (D_t, s_t) \tag{1}$$

Where, the state s_t, is used as input for the next time step predictions. The loss in this case is the same loss of YOLO3D however the optimization is back-propagated through time via the injected Convolutional LSTM layer to maintain the temporal information.
Figure 1, illustrates the frame stacking and Convolutional LSTM object detection architectures.

Because of the recurrency, the network generated values depend not only on the current frame at time t, but also on the previous m frames. On the other hand, the ground truth values correspond to the current frame. The ground truth values are defined as $v^{(t)}$, while the network generated values at time t are defined as $\hat{v}^{(t_m)}$, where t_m indicates the value generated based on input m frames, from time t back to time $t - m - 1$. The total loss is calculated as shown in Eq.(2).

$$
\begin{aligned}
L_\theta = \lambda_{coor} \sum_{i=0}^{s^2} \sum_{j=0}^{B} L_{ij}^{obj} \Big((x_i^{(t)} - \hat{x_i}^{(t_m)})^2 + (y_i^{(t)} - \hat{y_i}^{(t_m)})^2 + (z_i^{(t)} - \hat{z_i}^{(t_m)})^2 \Big) \\
+ \lambda_{coor} \sum_{i=0}^{s^2} \sum_{j=0}^{B} L_{ij}^{obj} \Big((\sqrt{w_i^{(t)}} - \sqrt{\hat{w_i}^{(t_m)}})^2 + (\sqrt{l_i^{(t)}} - \sqrt{\hat{l_i}^{(t_m)}})^2 + (\sqrt{h_i^{(t)}} - \sqrt{\hat{h_i}^{(t_m)}})^2 \Big) \\
+ \lambda_{yaw} \sum_{i=0}^{s^2} \sum_{j=0}^{B} L_{ij}^{obj} (\phi_i^{(t)} - \hat{\phi_i}^{(t_m)})^2 \\
+ \lambda_{obj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} L_{ij}^{obj} (C_i^{(t)} - \hat{C_i}^{(t_m)})^2 \\
+ \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} L_{ij}^{noobj} (C_i^{(t)} - \hat{C_i}^{(t_m)})^2 \\
+ \lambda_{classes} \sum_{i=0}^{s^2} \sum_{j=0}^{B} L_{ij}^{obj} \Big(- \sum_{c \in classes} p_i^{(t)}(c) log(\hat{p}_i^{(t_m)}(c)) \Big)
\end{aligned}
\tag{2}
$$

Where: λ_{coor} : the weight assigned to the loss over the coordinates, $\lambda_{obj}, \lambda_{noobj}$: the weights assigned to the loss over predicting the confidences for objects and no objects respectively, λ_{yaw} : is the weight assigned to the loss over the orientation angle, $\lambda_{classes}$: the weight assigned to the loss over the class probabilities, L_{ij}^{obj} : a variable that takes the value of 1 if there is a ground truth box in the ith cell and the jth anchor is the associated anchor, otherwise 0. L_{ij}^{noobj} : the complement of the previous variable, takes the value of 1 if there is no object, and 0 otherwise, x_i, y_i, z_i : the ground truth coordinates, $\hat{x_i}, \hat{y_i}, \hat{z_i}$: the predicted coordinates, $\phi_i, \hat{\phi_i}$: the ground truth and predicted orientation angle respectively, $C_i, \hat{C_i}$: the ground truth and predicted confidence respectively, w_i, l_i, h_i : the ground truth width, length, and height of the box respectively, $\hat{w_i}, \hat{l_i}, \hat{h_i}$: the predicted width, length, and height of the box respectively and $p_i(c), \hat{p}_i(c)$: the ground truth and predicted class probabilities respectively. B is the number of boxes, and s is the length of one of the sides of the square output grid, thus s^2 is the number of grids in the output.

4 Experimental setup

4.1 Dataset

All the experiments are conducted on the publicly available KITTI raw dataset [4], which consists of sequenced frames, unlike the KITTI benchmark dataset [4]. The dataset consists of 36 different annotated point cloud scenarios of variable lengths and a total of 12919 frames. These scenarios are divided into clips as described in section 3.2, with $m = 4$. Moreover, these scenarios have diverse driving environments such as highway roads, traffic, city and residential areas. They are also rich with dynamic and static objects.
We study the effect of Spatio-temporal object detection on 5 classes of objects. Our objects of interest were: Pedestrians, Cyclists, Cars, Vans, Trucks. 80% of the dataset is used for training, and the remaining 20% for evaluation. We choose not to ignore Vans or Trucks or devise a separate model for each class, as popular approaches in LiDAR-based object detection do [6][3]. In contrast, we are benefiting from YOLO as a single shot detector recognizing all KITTI classes in the same time, which makes our approach more practical for automated driving.

4.2 Bird Eye View (BEV)

The same point cloud pre-processing procedure in YOLO3D [1] is followed to generate the BEV input. In particular, we project the PCL into a single channel height grid map of size 608x608, at a cell resolution of 0.1m. The height grid map encodes the height of the highest PCL point associated with that cell. The major difference from YOLO3D's BEV is using a single channeled input instead of two, that is, we discarded the density channel, because we found that training with the height channel only did not significantly hurt the performance, while reducing the memory footprint, and allows more efficient training.

4.3 Architectures

Two fully convolutional object detection architectures are adopted for the experiments: YOLO-v2[18] in half precision, following [12] for a mixed precision training and Tiny-YOLO [17] in full precision [16]. We extended them for oriented 3D bounding boxes detection as mentioned in section 3.1, hence we call them Mixed-YOLO3D and Tiny-YOLO3D respectively. The motivation behind using mixed precision training and having the model weights in half precision is because that YOLO4D full precision model requires very high memory footprint to train effectively with a reasonable batch size. Tiny-YOLO [17] is adopted to experiment the Spatio-temporal effect on shallower models.

Frame Stacking: as described in section 3.2.1, each clip's m frames are stacked along the channel dimension and the input layer of Mixed-YOLO3D and Tiny-YOLO3D is modified accordingly.

Convolutional LSTM: as described in section 3.2.2, Mixed-YOLO3D and Tiny-YOLO3D networks are extended to account for the temporal dimension, by injecting a Convolutional LSTM layer just before the final output layer, revealing Mixed-YOLO4D and Tiny-YOLO4D respectively. This recurrent layer takes the $19 \times 19 \times 1024$ feature map, produced by the last convolutional hidden layer in both models as an input per time step, and outputs 512 channels using a 3×3 kernel size, which encodes the Spatio-temporal features that are fed to the final output layer for object detection and classification.

4.4 Training

For all Spatio-temporal based models, a clip length m of 4 is used. All models are trained till convergence, with a fixed batch size of 4, and a weight decay of $5e-5$. For optimization, scholastic gradient descent (SGD) is used with a momentum of 0.99, and a learning rate of $1e-4$. Regarding the half precision models: Mixed-YOLO3D, Mixed-YOLO4D and Frame Stacking + Mixed-YOLO3D, we followed [12] for a mixed precision training, forward pass in half precision, loss computation and weights update in full precision. No loss scaling is used for Mixed-YOLO3D and a loss scaling of 8 is used for the other 2 models.

4.5 Robustness

In real-world applications, it is important to test the detection performance in case of noisy sensory readings. For simplicity, noisy inputs are conducted by adding Gaussian noise to the BEV input frames at different scales g, then the detection performance is examined. Examples of inputs at different Gaussian noise scales are shown in Figure 2.

5 Results

The performance of frame stacking and YOLO4D models are compared to assess the effect of temporal information. As a baseline, Mixed-YOLO3D and Tiny-YOLO3D, are also compared. The F1 scores on the validation set are shown in Table 1. Based on the experimental results, the deeper Mixed-YOLO models show a better performance than the shallower Tiny-YOLO models. As expected, the YOLO4D models outperform the frame stacking models. Frame stacking encodes the temporal information only through the reshaping of inputs, while YOLO4D encodes the temporal information in a more natural way through the recurrent convolutional LSTM layer allowing a better propagation of temporal representations through time. YOLO4D models outperform all other

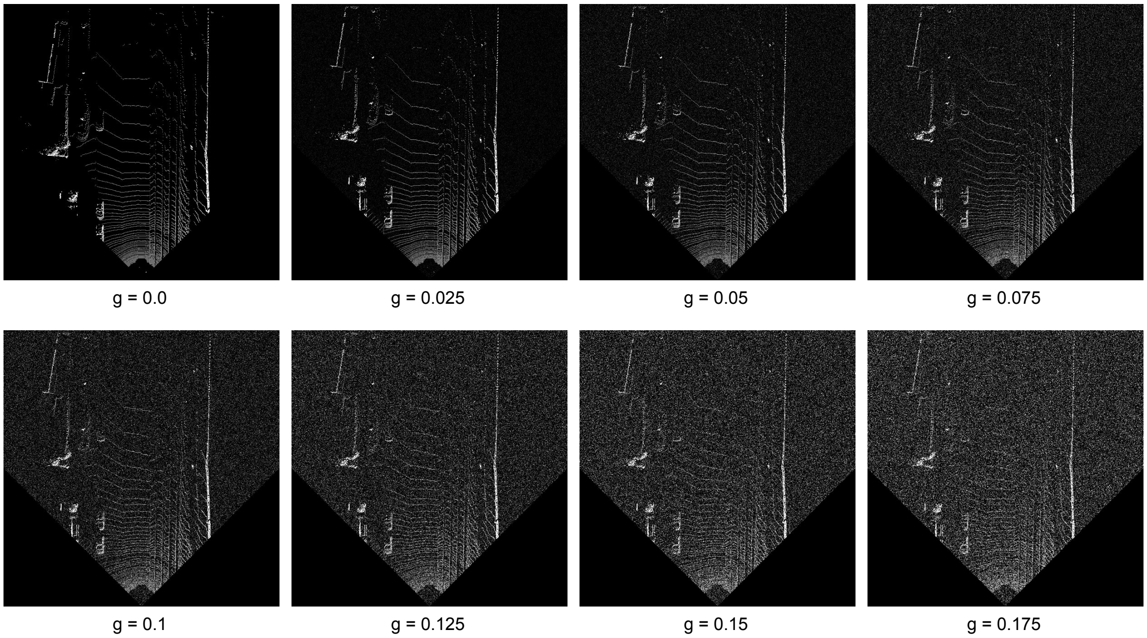

Figure 2: BEV inputs at different Gaussian noise scales g

Table 1: Performance Comparisons

Model	Mean F1 Score	Car	Pedestrian	Cyclist	Truck	Van
Mixed Precision:						
Mixed-YOLO3D	66.23%	69.82%	5.59%	14.61%	68.69%	64.32%
Frame Stacking + Mixed-YOLO3D	68.59%	71.77%	6.42%	16.93%	71.62%	65.45%
Mixed-YOLO4D	**77.73**%	**82.16**%	**10.44**%	**27.19**%	**81.77**%	**80.88**%
Tiny-YOLO:						
Tiny-YOLO3D	36.10%	37.77%	1.53%	2.29%	39.17%	36.26%
Frame Stacking + Tiny-YOLO3D	50.66%	53.69%	2.56%	2.86%	60.32%	43.13%
Tiny-YOLO4D	**70.36**%	**74.24**%	**10.03**%	**19.49**%	**73.00**%	**69.53**%

methods on all classes, achieving 11.5% absolute improvement on Mixed-YOLO3D, and 34.26% on Tiny-YOLO3D. Frame stacking provides a 2.36% absolute improvement on the Mixed precision baseline model, and a 14.56% absolute improvement on the shallower Tiny-YOLO baseline model.

As shown in Figure 3, YOLO4D models exhibit more robustness across different scales of noisy inputs. The performance of YOLO3D baselines models and frame stacking models drop significantly, while YOLO4D models maintain almost the same performance between noise scales of 0.025 and 0.075. The frame stacking and baseline models have comparable performances. As expected, Frame Stacking + Tiny-YOLO3D model shows slightly more robustness compared to the baseline Tiny-YOLO3D. Surprisingly, the baseline Mixed-YOLO3D show slightly better robustness than the Frame Stacking + Mixed-YOLO3D model.

6 Conclusion

In this work, YOLO4D is proposed for Spatio-temporal Real-time 3D Multi-object detection and classification from LiDAR point clouds, where the inputs are 4D tensors encoding the spatial 3D information and temporal information, and the outputs are the oriented 3D object bounding boxes information, together with the object class and confidence score. The experimental results show the effect of adding the temporal in addition to the spatial features for achieving better detection.

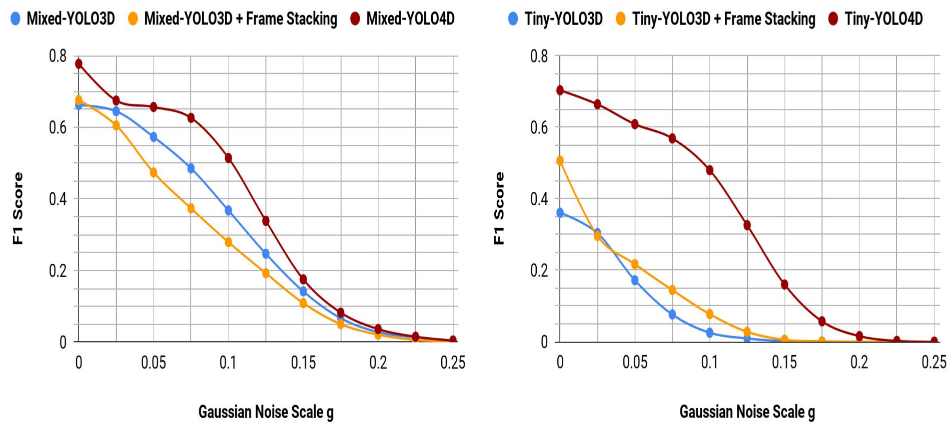

Figure 3: Robustness Performance: **Left**: Mixed-YOLO; **Right**: Tiny-YOLO

Recurrence and frame stacking are evaluated to incorporate the temporal dimension on KITTI dataset. Both recurrence and frame stacking show better detection performance compared to single frame detection. However, and as expected, recurrent YOLO4D achieves a better detection compared to frame stacking. Furthermore, robustness of the detection in the presence of noisy inputs is evaluated and it is clear that the YOLO4D models are more robust than frame stacking and single frame models.

References

[1] W. Ali, S. Abdelkarim, M. Zahran, M. Zidan, and A. E. Sallab. Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. *arXiv preprint arXiv:1808.02350*, 2018.

[2] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes. Depthcn: Vehicle detection using 3d-lidar and convnet. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 1–6. IEEE, 2017.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6526–6534. IEEE, 2017.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. Joint 3d proposal generation and object detection from view aggregation. *arXiv preprint arXiv:1712.02294*, 2017.

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

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

[9] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn. *arXiv preprint arXiv:1801.07791*, 2018.

[10] 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.

[11] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net.

[12] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaev, G. Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[13] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He. Spatially supervised recurrent convolutional neural networks for visual object tracking. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017.

[14] 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.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[16] 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.

[17] J. Redmon and A. Farhadi. Tiny yolo.

[18] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6517–6525. IEEE, 2017.

[19] S. Tripathi, Z. C. Lipton, S. Belongie, and T. Nguyen. Context matters: Refining object detection in video with recurrent neural networks. *arXiv preprint arXiv:1607.04648*, 2016.

[20] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[21] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds.

[22] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *arXiv preprint arXiv:1711.06396*, 2017.