Extensive Study of Camera-LiDAR Object Candidates Fusion for 3D Object Detection

Manthan Patel
Department of Mechanical & Industrial Engineering
University of Toronto
Toronto, Canada
manthan.patel@mail.utoronto.ca

Jigme Tsering
Institute for Aerospace Studies
University of Toronto
Toronto, Canada
jigmey.tsering@mail.utoronto.ca

***Abstract*—3D object detection tasks are usually performed based on multiple sensory data, such as LiDAR, image, and RADAR data. Image data usually provide us with extensive details on semantic features of objects present in the image, while LiDAR focuses on 3D localization and provides information on structural details of objects. Many efforts have been made on multimodal fusion to perform 3D object recognition, and many works have been mainly presented on 3 concepts of fusion early, intermediate and late fusion. Early and intermediate fusion are usually architecturally complex and require pixel-level correspondence. Meanwhile, the late fusion approach is quite convenient and simpler as it uses pre-trained single modality detectors to detect candidates, and then uses these candidates through fusion and processing to generate final detection candidates. We use the Camera-LiDAR Object Candidates' Fusion for 3D Object Detection (CLOCs) network and conduct a comprehensive study of the fusion architecture and detection framework. Our study is conducted using the KITTI training set and our test results were produced based on the KITTI validation set.**

I. Introduction

3D object detection is considered one of the most critical and difficult tasks in the field of computer vision, but the development of Deep Learning and CNNs has made this possible and has achieved quite good results in the last decade. There are many research works that deal with 3D object detection using monocular images, stereo images and LiDAR point clouds. However, each sensor has its own drawbacks. Image data can provide detailed semantic information about the objects present in the image, but it is difficult to extract accurate depth information from the image. LiDAR point clouds have accurate depth information and structural information about objects, but they lack semantic details. Currently, LiDAR-based detectors are very powerful, but they lack accurate depth and texture information. We could get more robust detections by using both image and LiDAR data. For this reason, several attempts have been made in recent years to fuse the modalities of the data to generate candidate 3D object detections [1].

Since we have multimodal data that can be fused to produce results, these fusion techniques have not been able to surpass the current state of the art in LiDAR-based 3D detectors. There are 3 types of fusion methods mainly based on the phase of data fusion: early fusion, intermediate fusion, and late fusion. Early fusion-based methods involve sequential processing of data. First, knowledge is extracted from the 2D detection or segmentation network and an attempt is made to pass it to the point cloud to improve it. This improved point cloud is then fed into the LiDAR detection network for detection [1]. Intermediate fusion is a bit more complicated. 3D object proposals are generated from the LiDAR-based detector, which are then projected onto multiple view types to crop features from the image or LiDAR backbone. Once we have cropped the features, they are fed into the final RoI pipeline to obtain object details.These methods are quite complex and computationally intensive, which is why the late- fusion methods were introduced in [1]. In the late- fusion methods, the output of two modality networks is used and fused at the end. Thus, we can omit the complicated fusion of intermediate features and modification of point clouds. Candidates' Fusion for 3D Object Detection (CLOCs) implements this idea of late fusion for 3D object detection [18].

The CLOCs network uses pre-trained 2D and 3D detectors for its image and LiDAR processing pipelines, respectively. It takes 2D and 3D detection candidates before the NMS and passes them to the fusion network to generate accurate 3D object parameters. This network is very flexible to implement and provides quite competitive results.

In this work, we attempt to implement CLOCs late fusion architecture and perform a comprehensive study on the late fusion technique. Our main motive for choosing the CLOCs network is the flexibility of the network and its versatility. Since the model performs late fusion, we can implement any 2D and 3D detectors to process image and LiDAR data, respectively. This gives us a variety of results that we can compare and study. Our main goal is to perform comparative studies on our different modifications in the CLOCs network and try to get better results if possible.

II. Related Work

Object detection is done using 3 methods: (1) 2D Image (2) 3D Point clouds (3) both Images and Point Clouds. 2D image-based methods usually give poor results compared to later methods.

A. 3D Detection Using 2D Image

Following the 2D detection approaches, a simple solution for monocular 3D object detection is to directly regress 3D

box parameters from images over a convolutional neural network. The direct regression methods naturally borrow from the architectures of the 2D detection networks and can be trained end-to-end. These approaches can be divided into single-stage/two-stage or anchor-based/anchor-free methods. These image-based methods are promising, but compared to LiDAR-based methods, however, produce much less accurate 3D bounding boxes.

B. 3D Detection Using Point Cloud

Currently, point cloud-based methods for 3D object Detection. They do not involve the complexity of multimodal fusion, such as calibration and synchronisation of multiple sensors. However, at longer distances, these methods do not provide good performance. These methods differ from each other in feature encoding techniques and learning methods. SECOND [2] is the improved version of [3]. Since the raw LiDAR point cloud has a very sparse data structure, sparse 3D CNNs are used, which significantly reduce the inference time. PointPillars [4] uses PointNets [5] in an encoder that represents point clouds in vertical columns (pillars) followed by a 2D CNN detection head to perform 3D object detection; it enables inference at 62 Hz; Compared to the single-stage methods discussed above, PointRCNN [6], Fast PointRCNN [7], and STD [8] use a two-stage architecture that first generates 3D suggestions in a bottom-up method and then refines these suggestions in a second stage. PV-RCNN in [9], the advantages of 3D voxel CNN and PointNet-based set abstraction are exploited to learn more discriminative features.

C. 3D Detection Using Multi-Model Fusion

To overcome the challenges of LiDAR-only detection methods, many papers describing multimodal data fusion have been published recently, such as image-LiDAR fusion, radar-LiDAR fusion, radar-camera fusion. The image LiDAR fusion method has provided promising results over time. Frustum PointNet [10], Pointfusion [11], and Frustum ConvNet [12] are representatives of 2D-guided 3D detectors that use mature 2D detectors to generate 2D proposals and narrow down the 3D processing area to the appropriate section in the image. However, 2D image-based proposal generation may fail in some cases that can only be observed in 3D space. MV3D [**?**] and AVOD [**?**] project the raw point cloud into the bird's eye view (BEV) to generate a multi-channel BEV image. Deep fusion networks are used to extract features from the BEV image and the 2D Camra image for bounding box regression. The overall performance of these fusion methods is inferior to that of the pure LiDAR-based methods. There is a possibility that due to multiple feature transformations and suggestions, images may lose their spatial information due to the destruction of the feature structure. Therefore, it is critical to develop a model that can use these features that can be fused together. The CLOCs architecture is one of the most convenient and flexible architectures because it uses recognition candidates from 2D image data and 3D point cloud data.

III. METHODOLOGY

A. Fusion of Detection Candidates

We take RGB images as input for 2D bounding box detection, the 2D detector gives us bounding box parameters and the confidence value of the object as output. While 3D detection systems produce classified oriented 3D bounding boxes with confidence values. We show only z-axis rotation in the results, while x- and y-axis rotation are set to zero. We use the calibration parameters of the KITTI dataset to visualize bounding boxes in LiDAR images. The main advantage of the early model fusion techniques is that they involve cross-modal fusion at an early stage to generate features, but at the same time, alignment issues, representation and sparsity are not well handled by the same network. Considering this drawback, intermediate fusion proposes fusion of modalities during feature processing, but this approach is still not convincing as it is not clear whether this increased complexity leads to real improvements.

Late fusion has advantages in training, as we can use individual modality algorithms to train their own sensor data. In the final fusion phase, matched and labeled data are merged. To weed out more detections, our aim is to maximize the recall rather than precision. For this reason, we are using detection candidates before the Non max suppression stage.

B. Consistencies

Since we fuse the detection candidates at the end, it is very important that the detections are consistent in the fusion. There are mainly 2 types of consistencies during fusion. Usually correctly detected objects have the same 2D and 3D bounding box parameters, while false positives are less likely to have the same bounding boxes. Therefore, we attempt here to quantify the geometric consistency between 2D and 3D detection. The detectors can output multiple object categories, but we associate only those belonging to the same category during the fusion. In this way, we obtain semantic consistency in our output.

C. Network Architecture

1) Sparse Input Tensor Representation: We are using same overall architecture proposed by [18]. The goal of the encoding step is to convert all individual 2D and 3D detection candidates into a set of all consistent common detection candidates that can be fed into our fusion network. The general output of a 2D object detector is a set of 2D bounding boxes in the image plane and the corresponding confidence values. For k 2D detection candidates in an image can be defined as follows:

$$\begin{aligned} \boldsymbol{P^{2D}} &= \{\boldsymbol{p_1^{2D}}, \boldsymbol{p_2^{2D}}, ..., \boldsymbol{p_k^{2D}}\}, \\ \boldsymbol{P_i^{2D}} &= \{[x_{i1}, y_{i1}, x_{i2}, y_{i2}], s_i^{2D}\} \end{aligned} \tag{1}$$

$\boldsymbol{P^{2D}}$ is the set of all k detection candidates in one image, for i_{th} detection $\boldsymbol{P_i^{2D}}$, x_{i1}, y_{i1} and x_{i2}, y_{i2} are the pixel coordinates of the top left and bottom right corner points from the 2D bounding box. s_i^{2D} is the confident score.

The output of 3D object detectors are 3D oriented bounding boxes in LiDAR coordinates and trusted scores. There are several ways to encode the 3D bounding boxes. In the KITTI dataset [15], a 7-digit vector is used that contains the 3D dimension (height, width, and length), 3D position (x,y,z), and rotation (yaw angle). For n 3D detection candidates in a LiDAR scan can be defined as follows:

$$\begin{aligned} \boldsymbol{P^{3D}} &= \{\boldsymbol{p_1^{3D}}, \boldsymbol{p_2^{3D}}, ..., \boldsymbol{p_k^{3D}}\}, \\ \boldsymbol{P_i^{3D}} &= \{[h_i, w_i, l_i, x_i, y_i, z_i, \theta_i], s_i^{3D}\} \end{aligned} \tag{2}$$

where $\boldsymbol{P^{3D}}$ is the set of all n detection candidates in one LiDAR scan, for i_{th} detection $\boldsymbol{P_i^{3D}}$, $[h_i, w_i, l_i, x_i, y_i, z_i, \theta_i]$ is the 7-digit vector for 3D bounding box. s_i^{3D} is the 3D confident score. Here 2D and 3D detection candidates which are being used are taken before non max suppression step, but still there are chances that some correct detections may be suppressed in single modality. But main motive is to use all candidates from both madalities to create predictions. So for k 2D detections, and n 3D detections we can have k x n x 4 tensor **T**, where we have 4 channels as follows:

$$\boldsymbol{T_{i,j}} = \{IoU_{i,j}, s_i^{2D}, s_j^{3D}, d_j\} \tag{3}$$

Here $IoU_{i,j}$ is IoU between i_{th} 2D detection and j_{th} 3D detection respectively. d_j is the normalised distance between j_{th} 3D bounding box and the LiDAR in xy plane. We will only consider elements with non-zero IoU in the tensor **T**, the rest will be zero valued. Thus, we can say that we will have overlapping 2D detections only for a few 3D detections. The rest of all elements in the tensor **T** are empty. We want to use only overlapping samples. So, to reduce the computational requirements and simplify the training process, we will remove non-zero elements from the tensor **T** and create a single 1 x p x 4 tensor, where p is the number of intersecting elements. Since we are using candidates prior to the NMS phase, the tensor **T** will be very large and contain as many elements as zero. Therefore, we now only use candidates that are non-zero and whose indices are cached. For 3D detections for which there is no corresponding 2D candidate, we still assign it an element in the tensor **T** with IoU and a confidence value of -1.

2) Fusion Network Details: In CLOCs [18] paper, a fusion network architecture as shown in Fig. 1 is proposed, but here we have prepared 3 new networks to perform a comprehensive study of different fusion networks. The input for these nets is a tensor of size 1 x p x 4 extracted from the sparse input tensor **T**.

a) Simple CNN Network: This is the basic network proposed in [18]. Simple 2D CNN layers as shown in Fig. 2(a) are used in this network. A total of 4 convolutional layers with a maximum of 36 channels are used. After each CNN layer, ReLU activation is applied. As the output of this network, we obtain a 1 x p x 1 tensor, which is converted to a sparse tensor using cached index data as before. We will then maxpool through the first dimension to obtain the desired probability value map of size 1 x n.

b) Complex CNN Network: This network as shown in Fig. 2(b) is our first modification of the simple CNN, keeping all parameters the same except for the channel sizes. We have a maximum of 64 channels, so our trainable parameters are increased. All layers are accompanied by ReLU activation layers. The network output is the same as the previous network. And we follow the same maxpooling procedure as mentioned before.

c) Branched CNN: In this network as shown in Fig. 2(c), we have created 2 branches of CNN network, one branch is more complex than the other. In one branch we have up to 64 channels with 3 CNN layers, while in another branch we have 16 channels with 3 CNN layers. We take the weighted average (by assigning more weight to the output of the complex branch) of the output of these layers and concatenate them at the end to produce an output of the form 1 x p x 1. We then follow the same procedure as before.

d) Encoder-decoder Type Network: The last network as shown in Fig. 2(d) we tried to create is an encoder-decoder type network, but the size of the embedding layer is larger than that of the input layer. At the end of each CNN layer, we added a ReLU activation layer and a dropout layer. After we get the output, the same steps as described before are performed.

D. Loss

We implemented the same loss function used in [18]. We use a cross entropy entropy loss for target classification, modified by the focal loss in [16] with parameters $\alpha = 0 : 25$ and $\gamma = 2$ to address the large class imbalance between targets and background.

E. Training

The CLOCs network is flexible and can incorporate any 2D or 3D detector combination into its pipeline. Thus, we used two 3D detectors, SECOND [2] and Voxel-RCNN, in our network. As a 2D detector, we have used Cascade RCNN, so we have presented a total of 2 combinations of 2D-3D detectors. For these two combinations, we have implemented all 4 fusion networks presented for study. The fusion network is trained using stochastic gradient descent (SGD) with an initial learning rate of 10^{-4} for 50 epochs.

IV. Results

A. Dataset

Our fusion system is evaluated on the challenging 3D object detection benchmark KITTI dataset [15] which has both LiDAR point clouds and camera images. There are 7481 training samples and 7518 testing samples. Ground Truth labels are only available for training samples. For experimental studies, we follow the convention in [17] to split the original training samples into 3712 training samples and 3769 validation samples. We compare our method with state-of-the-art multimodal fusion methods of 3D object detection on official test split of KITTI as well as validation split.

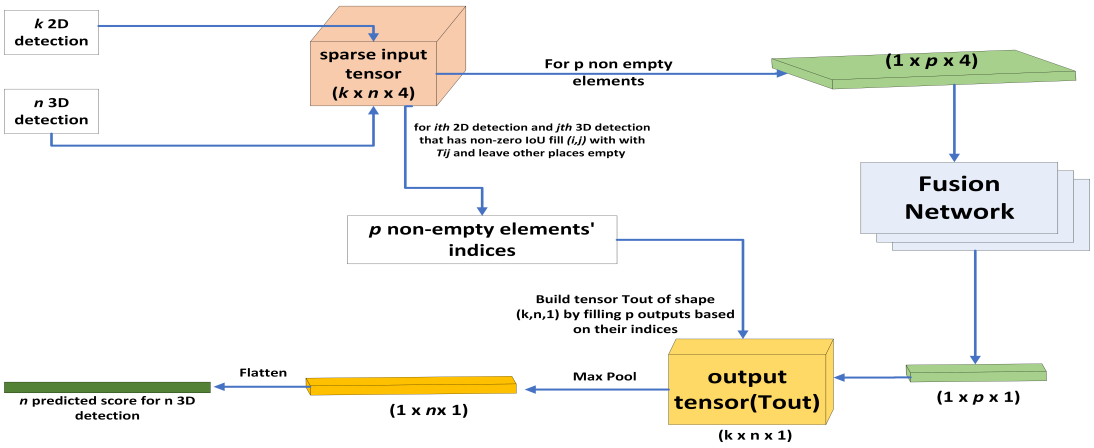
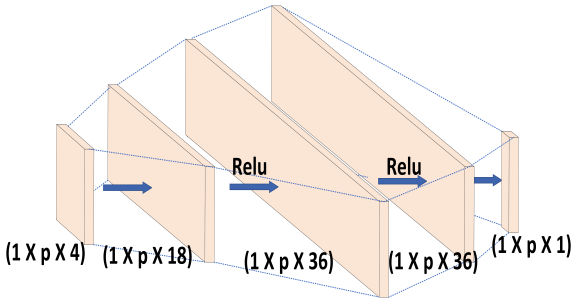
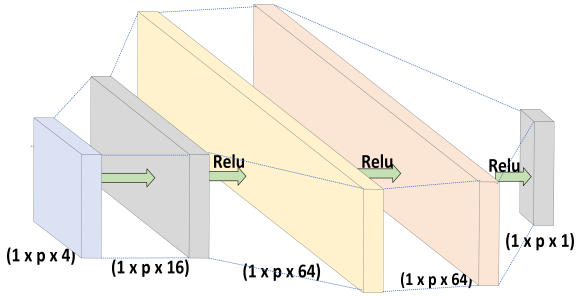


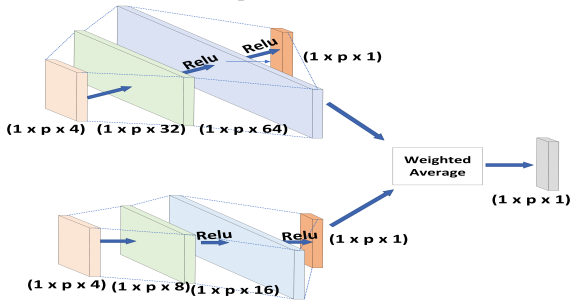
Fig. 1: CLOCs



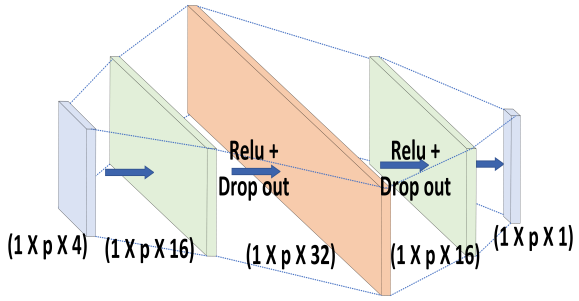
(a) Simple CNN network



(b) Complex CNN network



(c) Branched CNN



(d) Encoder-decoder type network

Fig. 2: Fusion network

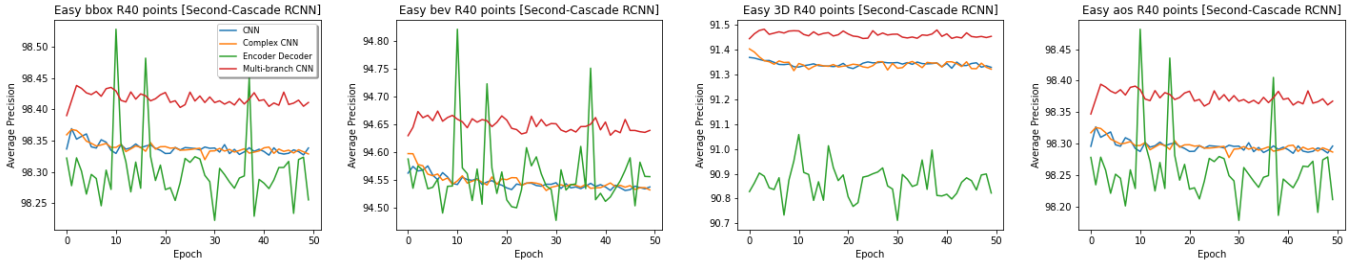


Fig. 3: Second-Cascade RCNN average precision

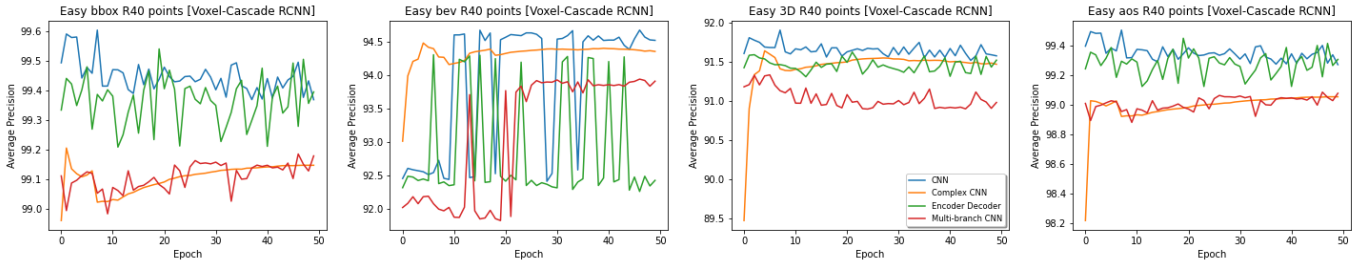


Fig. 4: Voxel-Cascade RCNN average precision

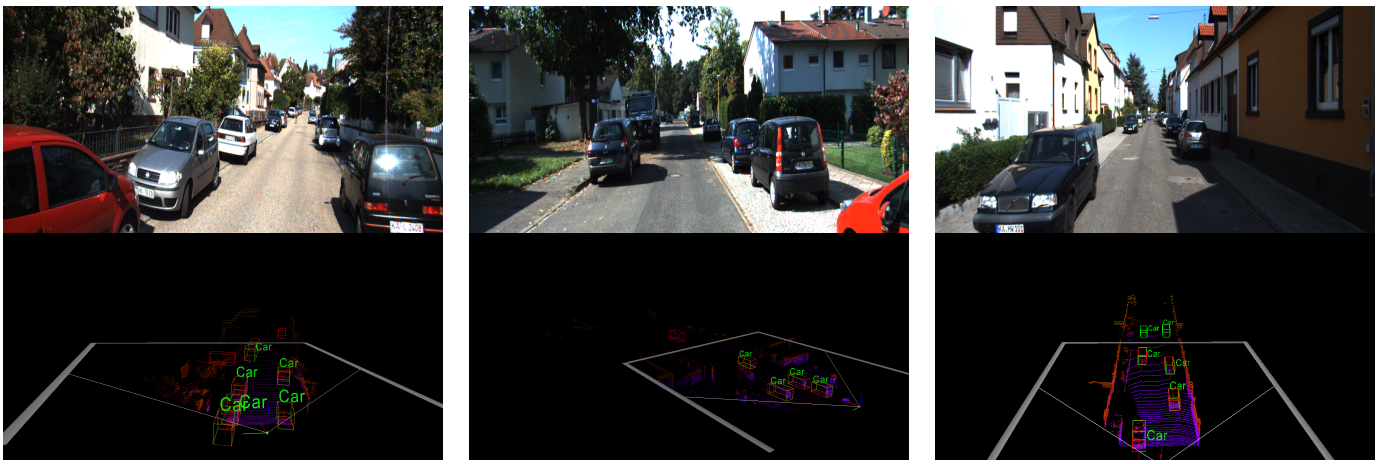


Fig. 5: Qualitative results of our CLOCs (Voxel RCNN- Cascade RCNN fusion) on KITTI validation set. Red and green bounding boxes are prediction and ground truth detections. The upper row in each image is the original image, the others are 3D detections in LiDAR point clouds.

B. Evaluation Results

We evaluated the detection results using the KITTI validation dataset. The IoU threshold for a car is 0.7, and all instances are classified into three difficulty levels: easy, medium, and hard, based on the height of the 2D bounding boxes, the degree of occlusion, and the degree of truncation. Two present our studies and evaluation results using training plots of average precesion, the table AP and qualitative results on sample images.

We plotted the plots of average precesion values for the easy difficulty level on recall 40 points. From the plots of SECOND -Cascade RCNN fusion, we can say that our modifications perform better than the actual fusion network proposed in CLOC's paper[638]. We plotted the values for Bounding Box, BEV, 3D bbox and Average orientation similarity's Average Precision for each epoch. We can say that our last modification of the encoder-decoder type of the CNN model performs better than all other models, with an increase of about 0.1 in the AP values. The CNN model with multiple branches also performs well and gives good results, as can be seen from the graphs.

In our second 2D-3D detector combination model of voxel RCNN-Cascade RCNN, the fusion network of the original model performs better. Compared to the original combination of SECOND and Cascade RCNN proposed in the paper, the values of our new combination AP are about +1.3 higher, which is a significant improvement.

From the LiDAR image visualization, we can see that the voxel RCNN-cascade RCNN fusion model performs better compared to the SECOND -cascade RCNN model. It provides better 3D bounding box predictions compared to SECOND -Cascade. Also, the model generates bounding boxes for partially or almost invisible objects that are not present in the ground truth, which we can see from images where green colored boxes are ground truths and red colored boxes are predictions.

TABLE I: 3D, bird's eye view, AOS, bounding box performance of fusion on car class of KITTI validation set (using new 40 recall positions) with Voxel-CasRCNN.

Score	Easy	Moderate	Hard
bbox AP	99.5846	94.5339	84.3932
bev AP	92.6020	87.8461	80.1837
3d AP	91.7866	84.1669	74.1887
aos AP	99.49	94.31	84.12

In the Table I and II we are comparing both our detection combinations for Recall 40 points. Here threshold values of IoU are considered as 0.7. We can see that in case of easy difficulty level our new setup of Voxel RCNN-Cascade RCNN

TABLE II: 3D, bird's eye view, AOS, bounding box performance of fusion on car class of KITTI validation set (using new 40 recall positions) with Second-CasRCNN.

Score	Easy	Moderate	Hard
bbox AP	98.5281	95.3344	92.5336
bev AP	94.8219	89.3970	86.6342
3d AP	91.0583	80.4093	77.2605
aos AP	98.48	95.11	92.22

performs better than the original one while it has comparable performance in case of moderate and hard difficulty level.

V. Future Scope

CLOCs model has a great level of flexibility so we can implement many modifications in it, though we could try only 2 2D-3D detector setups, but we can try to implement more setups. We are currently working on Yolo-v7 as 2d detector pipeline. Also access to more compute power can enable to try more variations in fusion network, also we can try making more complex network for fusion, which might result in giving us better results. We have already seen that our more complex fusion network gave us better results. Also, we can try using direct sparse input tensor as our input to fusion network instead of only taking non-zero elements.

VI. conclusion

From our study of CLOCs model we can say that due to great flexibility we can implement many modifications in this model. Though due to logistical constraints we could not implement all of them, we can perform better modifications and evaluations. Our implementations were successful and show better results than the original model. CLOCs also maintains the semantic and Geometric consistencies in detections and provide baseline for other early and deep fusion networks.

References

[1] Rui Qian, Xin Lai, Xirong Li, 3D Object Detection for Autonomous Driving: A Survey, Pattern Recognition, Volume 130, 2022, 108796, ISSN 0031-3203.

[2] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," Sensors, vol. 18, no. 10, p. 3337, 2018.

[3] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.

[4]] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[5] 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, 2017, pp. 652–660.

[6] 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, 2019, pp. 770–779.

[7]] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point r-cnn," in Proceedings of the IEEE international conference on computer vision (ICCV), 2019.

[8] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: sparse-to-dense 3d object detector for point cloud," ICCV, 2019. [Online]. Available: http://arxiv.org/abs/1907.10471.

[9] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pvrcnn: Point-voxel feature set abstraction for 3d object detection," in CVPR, 2020.

[10] 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, 2018, pp. 918–927.

[11] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 244–253.

[12] Z. Wang and K. Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection," in IROS. IEEE, 2019.

[13] 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, 2017, pp. 1907–1915.

[14] 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). IEEE, 2018, pp. 1–8.

[15] 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. IEEE, 2012, pp. 3354– 361.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll´ar, "Focal loss for dense object detection," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[17] 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, 2015, pp. 424–432.

[18] Pang S., Morris D., Radha H. (2020) Clocs: Camera-lidar object candidates fusion for 3d object detection. In: IROS.