2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Fusing Bird's Eye View LIDAR Point Cloud and Front View Camera Image for 3D Object Detection

Zining Wang, Wei Zhan and Masayoshi Tomizuka

Abstract— **We propose a new method for fusing LIDAR point cloud and camera-captured images in deep convolutional neural networks (CNN). The proposed method constructs a new layer called sparse non-homogeneous pooling layer to transform features between bird's eye view and front view. The sparse point cloud is used to construct the mapping between the two views. The pooling layer allows efficient fusion of the multi-view features at any stage of the network. This is favorable for 3D object detection using camera-LIDAR fusion for autonomous driving. A corresponding one-stage detector is designed and tested on the KITTI bird's eye view object detection dataset, which produces 3D bounding boxes from the bird's eye view map. The fusion method shows significant improvement on both speed and accuracy of the pedestrian detection over other fusion-based object detection networks.**

I. Introduction

LIDAR and camera are becoming standard sensors for self-driving cars and 3D object detection is an important part of perception in driving scenarios. 2D front view images from cameras provide rich texture descriptions of the surrounding, while depth is hard to obtain. On the other hand, 3D point cloud from LIDAR can provide accurate depth and reflection intensity information, but the resolution is comparatively low. Therefore, images and point cloud are complementary to accomplish accurate and robust perception. The fusion of these two sensors is a natural step for autonomous vehicles to deal with complicated driving scenarios.

The recent progress of convolutional neural networks (CNN) for classification and segmentation has invoked particular interest in applying deep neural networks (DNN) for object detection. The DNN-based object detection with either LIDAR [1], [2] or camera [3]–[5] has been widely explored by researchers and pushed to a very high single-frame accuracy. However, 3D object detection is still hard for networks based on a single kind of sensor. The camera-based network obtains high average precision on 2D image bounding box because of the rich texture information. Since a 2D camera does not contain accurate depth information, the 3D bounding box precision is very low for camera-based networks even with stereo vision [6] and prior knowledge of vehicle dimension [3]. Meanwhile, LIDAR-based networks have relatively low accuracy in the front view.

Sensor fusion gains the capability of achieving high accuracy in both 2D and 3D detections. Some fusion-based networks [7], [8] directly adapt from the Faster-RCNN structure with proposals from LIDAR so as to preserve the 3D information. [9] and [10] propose regions of interest from the front view but projects point cloud to the camera image plane for an early-stage fusion of information. The more recent MV3D [11] designs a complicated version of fusion after region proposal and gets the highest performance in 3D detection. All the networks to date do not fuse before 3D proposals of interested regions, which means that the fusion only helps the classification and regression stage. Fusion does not take place before highly potential objects are extracted by the region proposal.

In this paper, we propose a new camera and LIDAR fusion architecture in CNN that fuses different views before the region proposal stage. The fusion is allowed to happen before the proposal stage because it transforms the whole feature map instead of regions of interest(ROI), which is not presented in the previous fusion-based networks. The main idea is to employ the point cloud and sparse matrix multiplication to transform feature maps between different views, such as bird's eye view and front view, efficiently. The method is integrated into an innovative sparse non-homogeneous pooling layer. A fusion-based detection network is constructed accordingly. The single-sensor data processing units are modularized and are quite flexible. Therefore, the CNN backbones can be directly adapted from the state-of-the-art networks developed for single sensor. In this paper, we are able to switch among VGGnet used by [11], multi-scale network by [4] and 3D-CNN by [12], and benefit from the best single-sensor network.

Another contribution is that the new architecture fuses both LIDAR and camera information for 3D proposal, while other fusion-based networks merely use the LIDAR information. As a result, other networks have to fuse information after 3D regions proposals. When the fusion happens only after the region proposal part, the network is forced to use the old fashioned two-stage Faster-RCNN structure which is slow for real-time implementations. By fusing in an earlier stage, this paper gets rid of the region proposal subnet + detection subnet structure and benefits from the speed boost of recently developed one-stage detection networks like RetinaNet [13] and SSD [14].

II. Related Work

A. DNN for Object Detection

This section reviews the recent development of DNN-based detectors that are applicable to autonomous driving scenarios. One frame captured by sensors contains multiple

Z. Wang, W. Zhan and M. Tomizuka are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720 USA (e-mail: wangzining@berkeley.edu, wzhan@berkeley.edu, tomizuka@berkeley.edu).
*This work was supported by Berkeley DeepDrive (BDD).

objects involved in driving scenarios. The network slides a window or clusters dense points on the feature map to produce some regions of interest (ROIs). It is called the region proposal stage, also referred to as the first stage in detection. The ROIs are then fed to a classifier to generate classes and locations of objects, called the second stage. The most commonly used structure is Faster-RCNN.

The above two-stage network structure does not meet the time requirement in self-driving application. Recent works start to seek for one-stage detectors to avoid the overheads caused by the extra processing between two stages. SSD and YOLO [15] finish classification in the proposal stage and are much faster than two-stage detectors. As a trade-off, the accuracy trails due to class imbalance. The state-of-the-art RetinaNet addresses the problem with focal loss [13].

B. *Detector Based on Single Sensor*

DNNs based on a single sensor have been developed in many other areas. Detectors for images are very successful and their extensions to LIDAR are growing rapidly. Their CNN backbones serve as good candidates for the data processing units of a fusion-based network.

The camera-based detection receives the most attention from researchers. The competition is very intense on the KITTI dataset [16] with hundreds of proposed structures. Most well-performed camera-based networks including MS-CNN, RRC and Faster-RCNN only produce high quality bounding boxes in 2D front view images. There are surprisingly some single-camera-based networks capable of 3D detection. By adding the prior dimension knowledge of cars and cyclists, the earlier works such as 3DVP [17] and Mono3D [6] are able to produce 3D bounding boxes. The more recent works such as Deep3DBox [18] and DeepMANTA [3] produce even more accurate 3D boxes from monocular vision than networks using stereo vision.

The most intuitive thought of extending the successful 2D-CNN from images to the LIDAR point cloud is to use 3D-CNN with voxel representation. Unfortunately, the high computational complexity in the large outdoor scene makes it intractable. LIDAR data collected in autonomous driving scenario has its own inherent sparse property. Vote3Deep [2] conducts sparse 3D convolution to reduce computational load, but it does not apply to GPUs with parallel acceleration. VoxelNet [12] implements 3D convolution on voxel representation that is coarsely divided in height. It achieves highest scores in 3D detection with acceptable speed. VeloFCN [1] and SqueezeSeg [19] projects point cloud to the front view with cells gridded by LIDAR rotation and then applies normal 2D CNN for classification and segmentation. The front view representation of point cloud shares the same multi-scale problem as camera, because the sizes of objects change as distance varies.

C. *Fusion-Based Detector*

The network based on camera and LIDAR fusion, especially for pedestrian detection, has not been sufficiently investigated. Some works [9], [10] apply early fusion by projecting point cloud to the image plane and augment the image channels after upsampling. Such structure fuses camera and LIDAR data only on image plane. It means that the accurate 3D information of LIDAR point cloud is almost lost. The localization banks on that the regression can retrieve the 3D measurement of point cloud, which is not the case according to their relatively low 3D detection scores. Pose-RCNN [7] adapts the Fast-RCNN structure where the region proposal is done by classic selective search in LIDAR voxel representation. The fusion structure does not feed LIDAR data into the deep convolutional network which means that the classification relies merely on camera data. The state-of-the-art MV3D network applies region proposal network (RPN) on the point cloud projected to the bird's eye view plane. It preserves the 3D measurement in the region proposal stage. Then it uses the Faster-RCNN structure in the second stage. Note that the region proposal stage only takes the bird's eye view LIDAR data. The quality of proposals is limited by using single view and single sensor shown in Section V-B. The speed is also limited because the fusion must happen at the second stage. In this paper, we propose an efficient fusion scheme working in the first stage. The proposal takes into account the information from both CNN-processed bird's eye view LIDAR data and front view camera data. This structure not only produces high-quality proposals by camera and LIDAR fusion but also allows building fast one-stage fusion-based detectors.

III. SPARSE NON-HOMOGENEOUS POOLING

The sparse non-homogeneous pooling is a method that transforms two feature maps $\{f(x,y), g(u,v)\}$, where the transformation is linear but non-homogeneous. For example, the general convolution and pooling used in CNNs are homogeneous since

$$f\left(x,y\right)=\sum_{u,v}k\left(x-u,y-v\right)g\left(u,v\right)$$

The kernel $k(u,v)$ has the same finite support on (u,v), such as $[-1,1]\times[-1,1]$ with the kernel size of 3, which is independent of (x,y). The spatial transformer network [20] is also homogeneous as it predicts a uniform transformation matrix for the whole feature map. In general, the front view map and bird's eye view map are related by the projective transformation matrix $P\in\mathbb{R}^{3\times4}$ obtained from camera-LIDAR calibration. Denote (u,v) as the pixel in image and (x,y) as the coordinate in bird's eye view map. z is the additional height coordinate. If we want to transform a feature map from image to bird's eye view, the transformation is

$$f\left(x,y\right)=\sum_{u,v}k_{x,y}\left(u,v\right)g\left(u,v\right)$$

The support of kernel is

$$\sup\left(k_{x,y}\right)=\left\{(u,v)\,\middle|\,[u\ ,v]^T=w^{-1}P_{12}X, w\in\mathbb{R}^{+}\right\}$$

where $X=[x,y,z,1]^T$ and P_{12} contains the first two rows of P. Both the elements and measure of the support depend on (x,y). The support is usually a line. If the transformation was done as above, the computation would be heavy and non-parallel because a large but various number of $g(u,v)'s$ was involved for each (x,y). The proposed sparse non-homogeneous pooling uses the point cloud to shrink the support region. It also formulates the transformation as sparse matrix multiplication so that pooling of the full future map is done in one matrix multiplication.

A. Bird's Eye View LIDAR representation

The most important coordinate information of 3D object detection in autonomous driving scenarios is the x, y and orientation on the ground. In fact, the height of objects can be easily estimated from the ground. Bird's eye view takes the ground plane for the construction of feature map and thus provides the best proposal quality on x,y axis which are more difficult to estimate than the z axis. Currently, no front view detector achieves good performance in terms of 3D bounding boxes. In bird's eye view representation, point cloud is discretized into a $L_b \times W_b \times H_b$ grid. The $L_b \times W_b$ grid on the ground is dense and H_b is small so that the number of cells is similar to a 2D grid instead of 3D dense voxel representation.

There are two popular kinds of input features encoded in each cell. One is the hand-crafted feature such as density, height, reflection and position used in MV3D. The dimension is about 10 for each cell. Choosing what feature to use can be a difficult hyper-parameter tuning task. The other one is the network created feature by processing all points inside the cell, such as the PointNet structure[21] called voxel feature encoding (VFE) layer in VoxelNet. Each cell possesses a 128 dimensional feature learned by the network. In this paper, as we switch from different LIDAR data processing units, the input features are changed accordingly. As long as the features are arranged as gridded cells, the sparse non-homogeneous pooling method remains unchanged.

B. Non-homogeneous Pooling as Sparse Matrix Multiplication

The transformation between front view (image) map and bird's eye view map can be sparsified by the point cloud. Instead of matching one pixel (u,v) in front view with a full homogeneous line $(\lambda x, \lambda y)$ in bird's eye view, only (u,v) and (x,y) that share the same point in the point cloud are paired. The transformation is still non-homogeneous as the pairing does not guarantee one-to-n (n fixed) mapping, but the computation is sparse as the number of points in point cloud is of only 10,000 scale. Suppose the front view map is of size $H_f \times W_f$ and the bird's eye view map is $L_b \times W_b$. The LIDAR point cloud is $\{(x_i, y_i, z_i)\,|i = 1, 2, \cdots N\}$, the transformation kernel is sparsified as

$$k_{x,y}(u,v) = \delta_{(x,y)(x_i,y_i)} k_{x,y}(u,v)\, \delta_{(u,v)(u_j,v_j)}, \quad i,j = 1, 2 \cdots N$$

because only $(u,v),(x,y)$ correspond to a LIDAR point are transformed. The transformation becomes

$$\begin{aligned} f(x_i,y_i) &= \sum_{u,v} k_{x_i,y_i}(u,v)\, \delta_{(u,v)(u_j,v_j)} g(u,v) \\ &= \sum_{j} k_{x_i,y_i}(u_j,v_j)\, g(u_j,v_j) \\ &= \sum_{j} k_{x_i,y_i}(u_j,v_j)\, \delta_{(x_i,y_i)(x_j,y_j)} g(u_j,v_j) \end{aligned}$$

where

$$\delta_{ab} = \begin{cases} 1, a \sim b \\ 0, \text{otherwise} \end{cases}$$

So the transformation kernel, instead of having a small support like convolution, is a sparse $L_b W_b \times H_f W_f$ matrix of $\{(x,y)\} \times \{(u,v)\}$, or of $\mathbb{Z}_N / \sim \times \mathbb{Z}_N$ where $\sim$ is the equality and $[\cdot]$ means round to integer.

$$(x_i, y_i) \sim (x_j, y_j) \Leftrightarrow ([x_i], [y_i]) = ([x_j], [y_j])$$

There are at most N non-zero elements in the sparse matrix regardless of the size of feature maps. The kernel is normalized by the number of points in one cell.

$$k_{x_i,y_i}(u_j,v_j) = |\{k \in \{1, 2, \cdots N\}\,|(x_i,y_i) \sim (x_k,y_k)\}|^{-1}$$

Note this can be extended to more general interpolation methods like bilinear pooling by associating the LIDAR point with not only the one pixel it projects to, but also its neighbors and normalize the matrix row-wise to have sum 1.

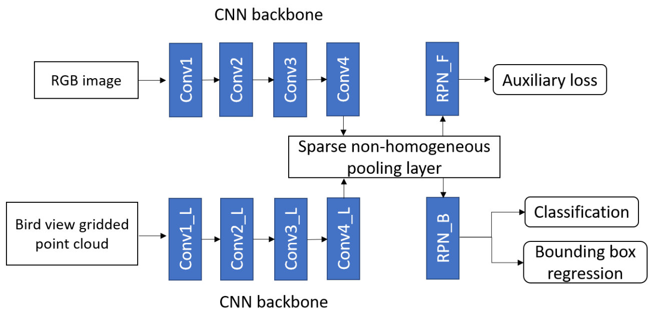

Fig. 1: The vanilla fusion-based one-stage object detection network.

C. Fusion of Front View and Bird's eye View Features

Section III-B introduces the case of transformation from front view (u,v) to bird's eye view (x,y) but this can also be done from bird's eye view to front view by just exchanging the coordinates. When applying the non-homogeneous pooling in network, since all the coordinate of the feature maps and the calibration matrix P are known, the pairs and matrix can be pre-calculated for each frame. There is no parameter to train for the pooling operation. The structure of the layer is illustrated in Figure 2

The sparse non-homogeneous pooling layer takes the feature map, such as the image, as input and $(u_i, v_i), (x_i, y_i)$ pairs as parameters. The layer constructs a $L_b W_b \times H_f W_f$

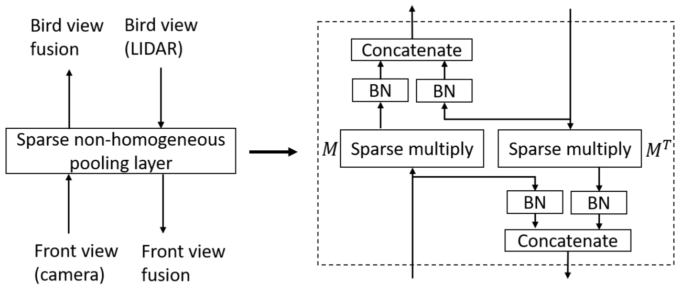

Fig. 2: The sparse non-homogeneous pooling layer that fuses front view image and bird's eye view LIDAR feature.

sparse matrix M. The $H_f \times W_f \times C$ feature map is flatten to a dense $H_f W_f \times C$ matrix F. Then the pooled feature map is derived as matrix multiplication $B = MF$ of size $L_b W_b \times C$ Finally, B can be concatenated with the target feature map for further processing. Batch normalization (BN) is applied for both feature maps before concatenation.

The fusion between bird's eye view and front view feature maps is not recommended in the early stage. Due to the low resolution of point cloud compared to the camera image, pooling of raw input results in a very low usage of image data. 20,000 points in the point cloud only results in 0.4% usage of pixels in raw RGB image in KITTI dataset. The sparse pooling is preferred to be applied to deeper layers like the network shown in Figure 1. When the pooling is done at the conv4 layer where the raw input is downsampled by 8, most of the front view and bird's eye view features are involved in fusion except for the part of sky with no laser reflection.

IV. One-Stage Fusion-Based Detection Network

The fusion structure introduced in Section III allows one-stage detector because the fusion is done in the first stage across different views, unlike [11] where fusion is done only after RoI pooling in the second stage. In this section we introduce an one-stage detector that takes front view camera image and bird's eye view LIDAR point cloud as input and produces 3D bounding box from bird's eye view without RoI pooling. The detection scheme is adapted from [13] but those of [14], [15] are also compatible.

A. 3D Region Proposal with Fusion structure

The network structure is shown in Figure 1. There are two fully convolutional backbones, namely the image unit and LIDAR unit. The sparse non-homogeneous pooling layer serves as the cross-bridge of two units to exchange information between sensors. The image unit uses the same RPN structure as recent camera-based one-stage detectors. Although the region proposal is not used during test, an auxiliary loss is still applied on the image CNN so that image features get supervision from the label in front view in addition to the that from the 3D proposal. It serves the similar functionality as the auxiliary loss in [11].

Since the region proposal is in the bird's eye view, objects of different distances are of the similar size if they are in the same category. The vehicles in KITTI dataset have bounding box size (l, w, h) of (4.0, 1.6, 1.6)m $\pm$ (0.6, 0.1, 0.2)m and pedestrians are of (0.9, 0.6, 1.6)m $\pm$ (0.2, 0.1, 0.2)m in the bird's eye view plane. There is no multi-scale issue hence the multi-scale structure in [11] and [22] are not needed.

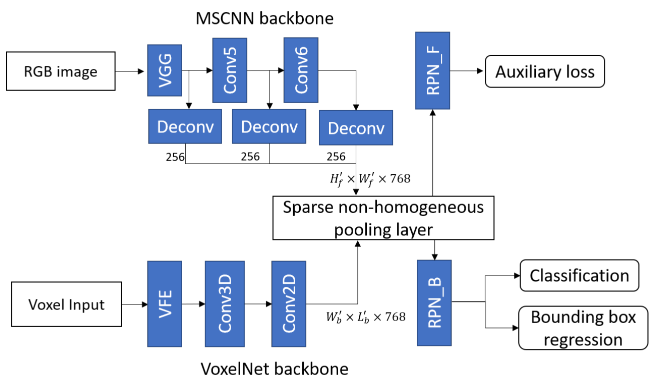

Fig. 3: The fusion-based one-stage object detection network with state-of-the-art single-sensor networks.

B. One-Stage Object Detection

The resolution of images and bird's eye view point cloud captured in the autonomous driving scenario is much larger than the benchmark datasets used to evaluate general detection networks. The test time per frame is important for practical application. One-stage detectors are much faster than two-stage detectors because they do not have the RoI pooling, non-minimum suppression and fully connected operation in the second stage.

One-stage detectors directly predict bounding boxes from all the proposals produced by RPN whose number is usually of 100K scale. However, there are only tens of objects so the positive and negative labels are highly imbalanced. The class imbalance problem in detectors and its solution in DNNs is summarized in [13]. For two-stage detectors [23], the imbalance problem is addressed by RoI pooling as it keeps a positive:negative$\approx$1:3 ratio in the second stage. Actually, other second-stage detectors like [11] also use bootstrapping and weighted class loss in the first stage when the class imbalance is huge for small objects.

Another interpretation of the class imbalance problem is that in addition to the prediction of probability, classification also involves binary (or discrete) decision [24]. A utility function or decision cost is imposed to the classification result where many true negatives, whose probabilities are below some threshold, contribute almost no cost. The so called hard negative mining solutions share the same idea of lifting the loss of hard negatives, because there is a mismatch between the used cross-entropy loss and the decision loss that actually measures the performance. This work uses the focal loss [13] which is a weighted sum of cross-entropy

$$\mathrm{FL}\left(\mathbf{p}, y\right) = \alpha_y (1 - p_y)^\gamma \mathrm{CE}\left(\mathbf{p}, y\right)$$

$\mathbf{p}$ is the vector of probability of the sample among all classes. y is the label of the sample. $p_y = \mathbf{p}(y)$ is the probability at class y and α_y is an empirical weight for each class. $\text{CE}(\cdot)$ is the cross-entropy. Focal loss is very efficient and is demonstrated to be effective on image-based one-stage object detector by [13].

V. Experiments

The network based on sparse non-homogeneous pooling is evaluated on the KITTI dataset which has calibrated camera and LIDAR data and ground truth 3D bounding boxes. The KITTI 3D object and bird's eye view evaluation are used. We focus on the pedestrian category as the maximum average precision (mAP) on KITTI is still very low among all fusion-based network where average precision (AP) is only about 26%. It is shown that the fusion structure helps a lot on the 3D proposal of pedestrians from bird's eye view since there is enough fused information.

A. Implementation

1) Network Details: Two kinds of CNN backbones are used for the network. The first one uses pretrained VGG for both LIDAR and camera, called the vanilla version shown in Figure 1. The input is the 1280×384 camera image and the $600 \times 600 \times 9$ gridded LIDAR bird's eye view representation with 0.1m resolution on the ground. The feature map produced by the backbone is down-sampled by 4 times in bird's eye view and 8 times in front view. The second one uses MS-CNN[4] for camera and VoxelNet[12] for LIDAR. The input is the 1280×384 camera image and LIDAR voxels following the same setting as VoxelNet. The feature map produced by the backbone is down-sampled by 2 times in bird's eye view and 8 times in front view. For both kinds, camera inputs are augmented by globally scaling randomly between $[0.95, 1.05]$ and shifting randomly between $[-10, 10]$. LIDAR inputs are augmented following the instruction of VoxelNet. The calibration matrices for sparse pooling is changed accordingly. The focal loss[13] is applied to all anchors to deal with the class imbalance.

2) Evaluation Speed: The sparse non-homogeneous pooling layer is tested to be efficient. The pooling of raw inputs (camera image and bird's eye view LIDAR above) takes 20ms while the pooling of features produced by VGG16 ($155 \times 46 \times 512$ for camera and $75 \times 75 \times 512$ for LIDAR) takes 14ms. With the efficient sparse non-homogeneous pooling fusion and one-stage detection structure, the test time is 0.11s per frame compared to the 0.7s per frame in MV3D. With VoxelNet, the inference time is longer because there is no available official version of VoxelNet. The implementation is modified from an unofficial publicly available reproduction by Jeasine Ma and does not reach the speed claimed by the paper [12]. The test time consists of the sparse matrix construction, network inference and bounding box post-processing (NMS) but the file I/O is excluded.

The network is evaluated on the KITTI object detection benchmark using the same train/validation split as provided in [4]. The bird's eye view average precision is used for evaluation. Actually the AP on bird's eye view benchmark is

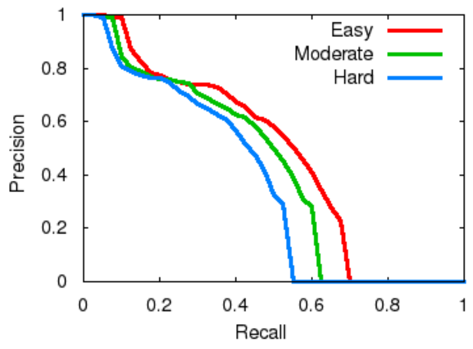

Fig. 4: The PR-curve of our fusion-based network on the validation set.

close to that on the 3D benchmark of KITTI for all methods involved.

B. Quality of Proposals

To verify the fusion structure is effective augmenting the feature in the RPN, the recall and precision in the RPN with VGG16 is compared with that of the MV3D structure. To make fair comparison, the cross-entropy loss is used in RPN instead of focal loss and IoU threshold is set to 0.5. The MV3D implemented in this paper is unofficial but reproduced according to the paper. Table I shows that for vehicles the precisions are similar but for pedestrians the fusion-based RPN has much higher precision. This is because the pedestrian is hard to distinguish with just LIDAR data, while vehicles are distinct from bird's eye view. The proposals for pedestrians require more information and the fusion structure provides the front view image feature, making proposal quality of pedestrians similar to that of vehicles.

Network	Vehicle		Pedestrian	
Type	Recall	Precision	Recall	Precision
MV3D	99.4	17.3	97.8	4.2
ours	99.2	19.5	96.6	17.3

TABLE I: RPN performance of RPN on pedestrian and vehicles on the validation set. All difficulty levels are included

C. Detection Results and Discussion

The pedestrian detection result is evaluated on the validation dataset and shown in Figure 4. Due to the limit of hardware, the network is not trained to the same number of epochs as in MV3D and VoxelNet which are around 150 epochs. The network is trained for 30 epochs on the train split with 3712 samples and evaluated on the validation split with 3769 samples. The learning rate is 0.0005 for the first 50% iterations and reduces linearly to 0 for the last 50%. The CNN backbone of LIDAR part is initialized randomly and the camera part loads the pretrained weights from VGG and MS-CNN, for vanilla version and VoxelNet+MS-CNN version respectively.

Table II shows the performance comparison of 3D detection networks presented on the KITTI dataset. The VoxelNet,

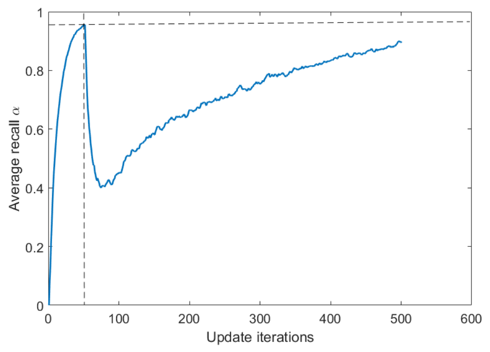

Fig. 5: The weight of focal loss within 30 epochs

due to its superior LIDAR input representation, has much higher score than other networks. It is even better than our vanilla version fusion-based network because the vanilla version uses the same hand-crafted LIDAR input as MV3D. By fusing VoxelNet and MS-CNN with the proposed sparse non-homogeneous pooling layer and one-stage detection network, we are able to achieve the highest performance on the validation set. Figure 5 shows the curve of average recall rate during training. The recall drops when the focal loss is activated and is still increasing at the end of the training process.

Network	Pedestrian		
Name	Easy	Moderate	Hard
3dssd	27.4	24.0	22.4
AVOD	34.4	26.1	24.2
VoxelNet(Official)	46.1	40.7	38.1
Vanilla	34.0	31.4	29.3
VoxelNet(Reproduced)	46.9	38.1	33.9
VoxelNet+MS-CNN	**51.3**	**45.0**	**40.2**

TABLE II: Comparison on the KITTI dataset. The last three rows are on the validation set with 30 epochs and the first three rows are on the test set with 150 epochs.

VI. Conclusion

We proposed a sparse non-homogeneous pooling method to efficiently transform and fuse features from different views of LIDAR point cloud and images from cameras. The proposed method enabled the fusion before the proposal stage so that the whole feature map can be exploited and fused, which was not presented in previous fusion-based networks. High-quality 3D proposals were produced by the corresponding one-stage detector designed. The detector achieved the best performance in the 3D detections of pedestrians from birds eye view on KITTI dataset, which was significantly superior to the state-of-the-art fusion-based detectors.

References

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

[2] 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 *Robotics and Automation (ICRA), 2017 IEEE International Conference*. IEEE, 2017, pp. 1355–1361.

[3] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," *arXiv preprint arXiv:1703.07570*, 2017.

[4] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision*. Springer, 2016, pp. 354–370.

[5] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," *arXiv preprint arXiv:1704.05776*, 2017.

[6] 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*, 2016, pp. 2147–2156.

[7] M. Braun, Q. Rao, Y. Wang, and F. Flohr, "Pose-rcnn: Joint object detection and pose estimation using 3d object proposals," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 1546–1551.

[8] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. Nunes, "Depthcn: Vehicle detection using 3d-lidar and convnet," in *Intelligent Transportation Systems (ITSC), IEEE 20th International Conference*. IEEE, 2017.

[9] T. Kim and J. Ghosh, "Robust detection of non-motorized road users using deep learning on optical and lidar data," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 271–276.

[10] S. Lange, F. Ulbrich, and D. Goehring, "Online vehicle detection using deep neural networks and lidar based preselected image patches," in *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE, 2016, pp. 954–959.

[11] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," *arXiv preprint arXiv:1611.07759*, 2016.

[12] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *CoRR*, vol. abs/1711.06396, 2017. [Online]. Available: http://arxiv.org/abs/1711.06396

[13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *arXiv preprint arXiv:1708.02002*, 2017.

[14] 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*. Springer, 2016, pp. 21–37.

[15] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[17] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1903–1911.

[18] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," *arXiv preprint arXiv:1612.00496*, 2016.

[19] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," *arXiv preprint arXiv:1710.07368*, 2017.

[20] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[21] A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez, "Pointnet: A 3d convolutional neural network for real-time object class recognition," in *International Joint Conference on Neural Networks*, 2016, pp. 1578–1584.

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *arXiv preprint arXiv:1612.03144*, 2016.

[23] 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*, 2015, pp. 91–99.

[24] F. E. Harrell Jr, *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015, ch. Prediction vs. Classification, pp. 4–5.