Pseudo-Image and Sparse Points: Vehicle Detection With 2D LiDAR Revisited by Deep Learning-Based Methods

Guang Chen, *Member, IEEE*, Fa Wang, Sanqing Qu, Kai Chen, Junwei Yu, Xiangyong Liu, Lu Xiong, and Alois Knoll, *Senior Member, IEEE*

***Abstract*— Detecting and locating surrounding vehicles robustly and efficiently are essential capabilities for autonomous vehicles. Existing solutions often rely on vision-based methods or 3D LiDAR-based methods. These methods are either too expensive in both sensor pricing (3D LiDAR) and computation (camera and 3D LiDAR) or less robust in resisting harsh environment changes (camera). In this work, we revisit the LiDAR based approaches for vehicle detection with a less expensive 2D LiDAR by utilizing modern deep learning approaches. We aim at filling in the gap as few previous works conclude an efficient and robust vehicle detection solution in a deep learning way in 2D. To this end, we propose a learning based method with the input of pseudo-images, named Cascade Pyramid Region Proposal Convolution Neural Network (Cascade Pyramid RCNN), and a hybrid learning method with the input of sparse points, named Hybrid Resnet Lite. Experiments are conducted with our newly 2D LiDAR vehicle dataset recorded in complex traffic environments. Results demonstrate that the Cascade Pyramid RCNN outperforms state-of-the-art methods in accuracy while the proposed Hybrid Resnet Lite provides superior performance of the speed and lightweight model by hybridizing learning based and non-learning based modules. As few previous works conclude an efficient and robust vehicle detection solution with 2D LiDAR, our research fills in this gap and illustrates that even with limited sensing source from a 2D LiDAR, detecting obstacles like vehicles efficiently and robustly is still achievable.**

***Index Terms*— Vehicle detection, 2D LiDAR, autonomous driving, deep learning, intelligent transportation system.**

Manuscript received May 15, 2019; revised October 19, 2019 and May 2, 2020; accepted June 26, 2020. Date of publication July 29, 2020; date of current version November 29, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61906138, in part by the State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body Open Project under Grant 31815005, in part by the European Union's Horizon 2020 Framework Program for Research and Innovation (Human Brain Project SGA3) under Grant 945539, and in part by the Shanghai AI Innovation Development Program 2018. The Associate Editor for this article was J. M. Alvarez. *(Guang Chen and Fa Wang contributed equally to this work.) (Corresponding author: Guang Chen.)*

Guang Chen is with the School of Automotive Studies, Tongji University, Shanghai 200092, China, also with the State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Changsha 410082, China, and also with the Department of Informatics, Technical University of Munich, 80333 Munich, Germany (e-mail: guangchen@tongji.edu.cn).

Fa Wang, Sanqing Qu, Kai Chen, Junwei Yu, Xiangyong Liu, and Lu Xiong are with the School of Automotive Studies, Tongji University, Shanghai 200092, China.

Alois Knoll is with the Department of Informatics, Technical University of Munich, 80333 Munich, Germany.

Digital Object Identifier 10.1109/TITS.2020.3007631

I. Introduction

VEHICLE detection is one of the most crucial tasks in many autonomous mobile applications, such as driverless cars, auto-parking robots, forward collision warning, and other ADAS functionalities. Since AlexNet [1], inspired by cognitive science [2], [3], deep learning has made great progress in object recognition, detection and semantic segmentation [4]–[8]. Currently, many systems and methods based on multiple laser beam sensors or visual sensors have been developed for vehicle and obstacle detection [9]–[12]. However, due to the considerable cost of a multi-layer LiDAR and the inaccuracy in distance measuring of vision sensors, it still remains a demand for more accessible and robust methods which are based on cheaper and more compact sized LiDARs with minimum laser beams.

The LiDAR systems usually provide data at a refresh rate of 10Hz to 50Hz and can measure up to a range of 50m to 120m. The range data is considered the most accurate measurement of distance among existing sensors. A 2D LiDAR sensor with two-dimensional laser beam is less expensive both in terms of purchasing price and computation capability requirements. Especially when considering that a widely used typical 3D LiDAR, Velodyne HDL-64E, alone produces up to 75MB point cloud data every second and still need to be processed in real time, which would be a heavy burden for onboard systems and almost impossible for most of embedded systems.

In addition to that, even the most high-class and cutting-edge 3D LiDAR alone can not cover the entire surrounding area, which means there must be critical blind spots near the ego-vehicle that require supplementary sensory while still with accurate ranging. In particular, radars or ultrasonic radars produce inaccurate measurements that would downgrade the overall quality of fusion decision making and may either cause unpredictable movements or false alarms. Therefore, that is yet another place where 2D LiDARs could take over at manageable costs.

Meanwhile in vision based perception, as mentioned before, target distance estimation from a passive sensor like a monocular or binocular relies heavily on sensor calibration and environment illumination, which makes it less accurate and robust. On the other hand, active types like a kinect RGB-D is very insufficient in detecting range.

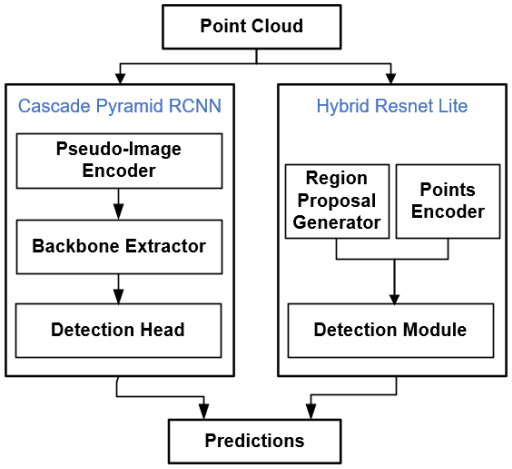


Fig. 1. The framework of proposed methods, Cascade Pyramid RCNN and Hybrid ResNet Lite. The former converts point clouds into encoded pseudo-images and learns features from cascade pyramids, while the latter directly proposes regions of interest (RoIs) using a non-learning algorithm before a light network performing the detection.

To address the above challenges from 3D LiDAR based and vision based methods, we revisit the vehicle detection problem with a less expensive 2D LiDAR by utilizing modern deep learning approaches. We aim at filling in the gap as few previous work concludes an efficient and robust vehicle detection solution in a deep learning way in 2D. To this end, we propose a learning based method with input of pseudo-images, named Cascade Pyramid RCNN, and a hybrid learning method with input of sparse points, named Hybrid Resnet Lite. The framework is illustrated in Fig 1.

The Cascade Pyramid RCNN is constructed based on fully end-to-end convolution network that takes in only 2D LiDAR frames. The network mainly consists of three major modules, the pseudo-image encoder, backbone extractor and the detection head. The first module encodes point clouds into unified tensors, while the other two form and learn across cascade feature map pyramids, fusing cues of targets of various sparseness and scales. The Hybrid Resnet Lite directly processes the data collected from 2D-LiDAR in forms of sparse points instead of pseudo-images. Before the detection module conducts the prediction, each point is re-encoded, then a non-learning algorithm proposes RoIs from these re-encoded sparse points.

In summary, we fill in the gap as few previous work concludes an efficient and robust vehicle detection solution in a deep learning way in 2D. Our main contribution lies in:

- By addressing the challenges of 3D LiDAR based and vision based methods, we propose an alternative for robust and efficient vehicle detection method based on less expensive 2D LiDAR.
- We revisit the 2D LiDAR based vehicle detection by developing two modern deep learning based methods: one focuses on performance by constructing cascade pyramid architectures and the other pursues speed and light weight by hybridizing learning based and non-learning based modules.
- To the best of our knowledge, we build the first dataset dedicated to vehicle detection in 2D range data for learning based methods. Our proposed Cascade Pyramid RCNN outperforms existing methods using the new dataset.[1]

II. Related Work

A. 2D LiDAR Based Detection

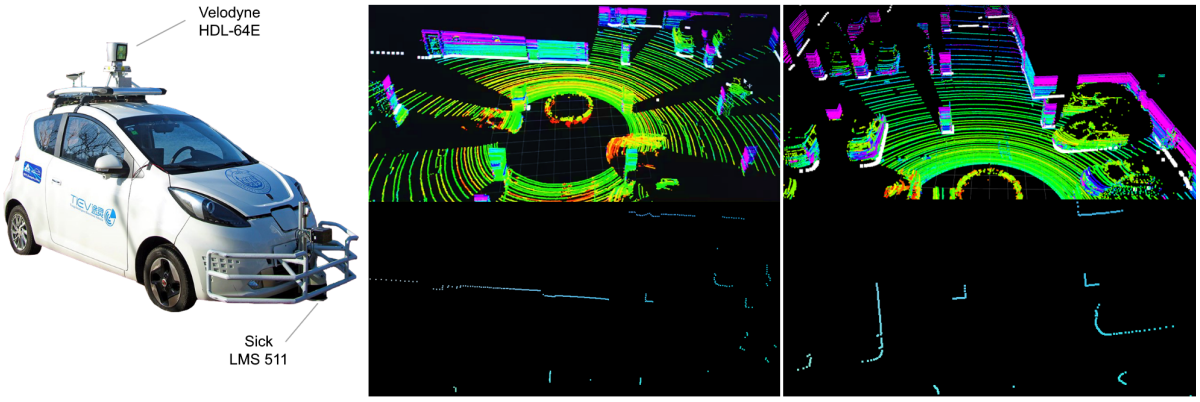
Despite the advent of 3D LiDAR sensors comes with richer perception information in three-dimensional space, horizontally mounted 2D laser scanners are still a very common part of sensory setups on mobile robot systems. There have been several research works trying to enhance the performance based on that. Present methods can be mainly divided into two categories. One is the conventional hand-crafted feature extraction from clustering with later machine learning techniques to do the classification and regression. Some focus on human detection and localization [13]–[16]. While some others seek to find solutions in detecting vehicles [17]–[23]. Among these, some pre-designed heuristics can be the cues to look for, like the geometric shape of legs, L-shape of cars or numerical value of laser points' medians or means. But such pre-designed-by-hand features can be easily obscured and confusing in many real-world scenarios due to occlusions or field of view limitations.

The other one is the new-fashioned end-to-end deep learning methods which can learn the optimal features from a large amount of well-annotated data and output targets positions without human intervene. Some works on detecting surrounding humans or wheelchairs based on convolutional networks have been proposed [24]–[26]. The basic neural network architectures are largely similar with those in vision perception uses [27], [28]. Though this paradigm requires a larger amount of data with diversity, the learned features are shown to be more effective and more robust in such tasks. Note that no previous deep learning method aiming at vehicle detection with 2D LiDARs only, partly because of the field of view limitation of 2D LiDARs and the rise of 3D LiDARs such as Velodyne HDL-64E. However, in particular scenarios, 2D LiDARs still hold their places. For example, in an auto-parking scenario, the parking robot needs range measurement from multiple directions and thus unrealistic to be equipped with more beams but way more expensive 3D LiDARs in every direction, especially considering mass production. In addition to that, even the most high-class 3D LiDAR alone can not cover the whole surrounding area, which means there must be blind spots that require supplementary sensory while still with accurate ranging (not like radars or ultrasonic radars which produce inaccurate measurements that would downgrade the quality of fusion decision making and may either cause unpredictable danger or false alarms), and that's where 2D LiDARs could take over at manageable costs.

B. 3D LiDAR Based Detection

With the difficulty of obtaining enough various and well-annotated point cloud data and dealing with data vectorizedity,

[1] https://github.com/ispc-lab/2DLiDAR-VehicleDetection



(a) (b) (c)

Fig. 2. (a) Data acquisition platform: the autonomous driving platform *TiEV* with a 3D LiDAR on the car's top and 2D LiDAR in the front. (b)-(c) The first row shows 3d and 2d point clouds where white points are from 2D LiDAR and colored points from 3D LiDAR representing different intensity. Birds eye views of corresponding 2D data are in the second row.

object detection in 3-dimensional point cloud is significantly more complicated than those within a 2D plane. Intuitionally, 3D convolutional network is the most straight forward way to be deployed in detecting architectures [29]–[31]. But 3D convolution is way slower and memory consuming, thus less state-of-the-art models are developed based on that.

The mainstream methods mainly convert the 3D point cloud into 2D images by projecting into birds-eye view plane or front view plane [32]–[34], where each pixel of the projected image encodes the information of points data in their corresponding grids. Then the converted pseudo-image can be processed following 2D image procedures. Some of these take in just point cloud [35], [36], while some others utilize fusion method from vision [37]–[39].

Recently, as Pointnet [40], [41] was proposed and proved to be effective in extracting features directly from raw point cloud, several models have been developed based on that [41]–[44], while the most successful ones among these methods still somehow convert the point cloud into a pseudo-image [45], [46], which is a dense and unified tensor compared with sparse and unordered points, and then proceed with standard 2D image detection architectures. During these conversions, learned features are, not surprisingly, showing better performance over fixed handcrafted ones.

However, as 2D laser scan points do not have such data vectorizedity to encode in a grid, usually with only a few hundred points in a grid, converting raw data to a pseudo-image is more straightforward.

III. Data Acquisition

A. System Setup

Our data collecting platform is a refitted electric vehicle in Fig 2 that is capable of fully autonomous driving named Tongji intelligent EV, i.e., TiEV [47]. TiEV is equipped with several types of sensors. Among these, a Sick LMS511 2D laser scanner is installed to cover the front blind spot left by a multi-beam HDL64 on the car's top. A differential global positioning system (DGPS) and an inertial measurement unit (IMU) system which consist of Novatel simpak6 receiver and Oxts RT2000 GNSS provide accurate ego-motion and global localization. All sensors have been carefully calibrated and systemically integrated.

The LMS511 laser scanner is mounted at approximately 18cm above the ground plane in front of TiEV's car body, with a camera behind the windshield and a Velodyne HDL-64E multibeam LiDAR on the car's top. Data from the GNSS and the IMU can be used for ego-motion compensation. The 2D scanner scans at 50Hz with a span of 190° at an angle resolution of 0.48°, totaling 391 measurements per scan with ten echoes. Scans start from the rightmost end to the left.

B. Dataset

1) Data Format: We use the Robot Operating System (ROS) to communicate between sensors and IPC (Industrial Personal Computer) and therefore ROS bags to record data. Point cloud data are given in the format of rosbag replays. Each bag file contains header topic of timestamps and diagnostic info, 3D point cloud from a Velodyne 64E-S2 as topic velodyne_points, 2D laser scan points from a LMS-511-10100PRO as topic sick_scan, ego-motion and localization info from Oxts RT2000 as topics imu and GPS, also the coordinates transformations between sensors and vehicle base_link as tf topics. The total number of bags is 31.

2) Recorded Scenes: We gather data from 9 different scenarios in 31 situation cases as shown in Fig 3 Fig 4 and Fig 5. Each case is recorded in one rosbag file. Various driving scenarios are taken in, including natural and artificial scenes and are mainly captured from 9 types of scenes, with about a total span of 1466.3 seconds, size of 132395.2 MB and msgs number of 1312034 as in Table I. These scenes are:

a) Static scenes: Static scenes are mainly collected from off-road parked cars around the campus site. Parking lots, underground parking garage, roadsides, building front yard and deserted stations with cars of different models and parked in unified or arbitrary heading directions. The data collecting vehicle passes along at different distances. To add to the

Fig. 3. Examples of data collecting Scenes. (a) and (d) are campus road scenes, (b) is country road, (c) is underground garage. Video frames (first row) are from a front camera recorder for annotation purpose, 2D point cloud in light blue with annotations of yellow bounding boxes are shown in birds eye view (second row).

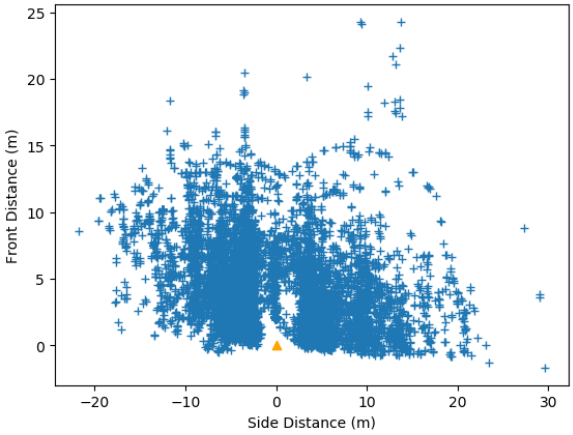


Fig. 4. Relative position distribution of targets' center points. Each blue cross represents a target observed from the sensor. The yellow triangle is the observing sensor.

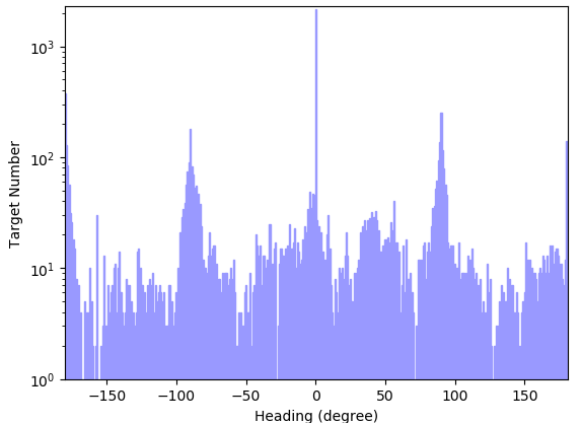


Fig. 5. Heading angle distribution of targets relative to the ego-vehicle's heading. Cars with the same heading of the ego-vehicle (0°) outnumber others.

variety, occlusions like columns, trees, walls, pedestrians or other cars, etc. are all recorded to reflect the realness in the environment.

b) Dynamic scenes: Dynamic scenes are mainly about real road driving scenarios, including campus roads, expressways, country roads, and city roads. There are intersections, bends, ramps, slopes. In real traffic, we record situations like counter car flow, making turnings, turning around and waiting for traffic lights. Occlusions involve isolation strips, bushes, and other barriers appeared along the road.

3) Annotation: We annotate every five frames in a frame batch per second, which originally consists of 50 frames, i.e. 10% of total frames are annotated. By doing so, we are still able to cover the whole dataset's information. In total, we annotated 10522 frames of lidar scans. And about half of them, 4240 annotated frames are with foreground targets. The average annotation box size is 1.3m in width, 2.4m in length. We utilized an under-development open sourced point cloud annotator based on ROS and RVIZ. The process is as follow: The data are first down-sampled to 5 Hz to keep synchronization of different data sources, then converted into .pcd format files for hand labeling according to the point cloud replays in RVIZ. The final annotations are given in plain text files each corresponds to its frame file, and each row [C,o,l,ω,h,xc,yc,zc,θ,α] stands for a visible target box with its class C, occlusion i, size in length, width, height, center coordinates of xc, yc, zc, target directions θ, and target heading direction α. To utilize the annotation for higher time resolution, interpolation can be further implemented because adjacent frames are only with minor changes.

IV. Methods

We propose two different methods based on deep learning techniques. Cascade Pyramid RCNN focuses more on performance while still maintains real time capability. Hybrid Resnet Lite pursues faster speed and lighter weight.

A. Cascade Pyramid RCNN

Targets in a 2d point cloud frame can be very sparse in space and various in scales. To this end, we propose Cascade Pyramid RCNN which utilizes a two-stage Region proposal Convolution Neural Network (RCNN) [48] with stacked feature map structured as in Fig. 6, namely cascade pyramids. Network architecture and pipeline are illustrated in Fig. 7.

TABLE I
DATASET OVERVIEW

ROSbag replay number	Scenario types	Total Duration (s)	Average Duration (s)	Annotated Frames	Foreground Frames
31	9	1466.3	47.3	10522	4240

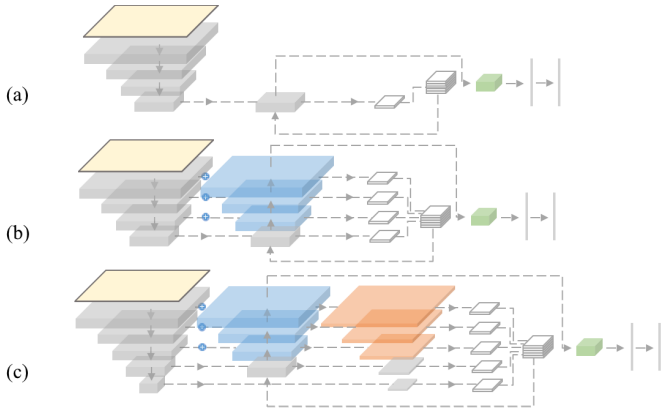


Fig. 6. Cascade Pyramid Disassembled. (a) Faster RCNN, (b) and (c) are RCNNs with different feature map pyramids added to the pipelines. (b) and (c) both are intermediate states in the tear-down comparison between our proposed model and normal Faster-RCNN. Compared with (c), (b) structure lacks the third pyramid.

1) Point Cloud to Pseudo-Image: First, we project points in every frame onto the ground plane with a span of 33.33 meters and grid resolution of 5.55 cm. Grids with no point will be applied with zero paddings. Next, each grid is encoded by the points coordinates x, y and distance to view point d where $d = \sqrt{x^2+y^2}$ as its RGB channel values. It will then be enhanced by its eight neighbors with bi-linear interpolation. The frames are later normalized and augmented by shuffling, and random flipping. After this process, the sparse, unordered and illy shaped point clouds are transformed into CNN-friendly unified and dense tensors, and therefore pseudo-images.

2) Feature Map Extractor: As shown in the left part of Fig.7, pseudo images are extracted to form feature map pyramids. Through these pyramids, regions of interest (RoIs) are proposed like in [5] by learning from offsets and errors between proposal boxes and matched ground truths. These RoI box proposals are then sent to next module.

To be more specific, in the first feature map pyramid structure, We use a residual network of 50 layers like in [6] as the backbone feature extractor over pseudo-images. The process is as follow. Through every couple of conv layers, the pseudo-image is down-sampled in size, and gradually extended in channels to reach 256. Five specific sized feature maps among them are collected together, forming a five-layer top-down feature map pyramid at increasingly small spatial resolution.

The second feature map pyramid is constructed reversely. The second last feature map is then passed through another conv layer before upsampled three times to build up this four-layer feature map pyramid. Note that at each up-sampled layer, the feature map is added by the corresponding feature map from the previous pyramid.

And the third pyramid composes of feature maps from the previous two. The second pyramid passes through conv layer in each level, providing four levels of feature maps. The last and the smallest feature map is from the first pyramid through another conv layer.

In this way, cues of different scale features can be better combined, for later, in the third pyramid, anchor boxes will be generated simultaneously and independently within each layer, capturing potential targets of different sizes and scales. Anchors that are matched successfully with ground truths will guide the network to produce better-fitted proposals boxes by learning from their offsets and errors in binary classification, box regression and box orientation. Proposals are then concatenated together, Non-Maximum Suppressed (NMS) [4], and re-organized by randomly sampling to keep foreground background balance in ratio.

3) Detection Head: The fourth pyramid structure is used in RoI detection head. Each proposal box cuts out its RoI feature map from the second pyramid, together forming yet another four-layers pyramid with smaller sizes. RoI align [7] is then conducted to further downsample the feature maps in different layers to a same size of 7×7. Notably, because during RoI head detection, the variation in target scales is less significant compared with those in the last module, proposals are assumed to already approximate the scale of their corresponding targets, only four levels of feature maps are used. Final detection of every target proceeds upon the aggregated RoIs. Fully connected layers flatten the last feature map and output the vector encoding desired results. The output will be overlaid back onto the original pseudo-image, and NMS-ed before final losses of classification, box regression, and orientation can be calculated and backpropagated.

4) Vectorized Angle Representation: As previous works in 2d LiDAR detection using deep learning usually did not achieve regression of target headings, we also demonstrate that the network is capable of regressing relatively accurate target heading.

Vectorized angle representation is introduced to overcome the mathematical singularity of regressing single angular value when estimating heading directions. If we define the direction angle value ∼[-pi, pi], a difference of around 2pi in angle would be falsely considered the largest in loss calculation and therefore lead to confusing gradient descent during learning, pushing the network away from potential optimum. As a result, this truncated value can confuse the network and cause unstable learning process.

To get rid of that, regression of two values representing an angle vector in its Cartesian coordinates instead of a single radian, i.e., one for the vector's projection on x axis and another on y axis would naturally avoid any truncation or singularity because they can be regarded as continuous

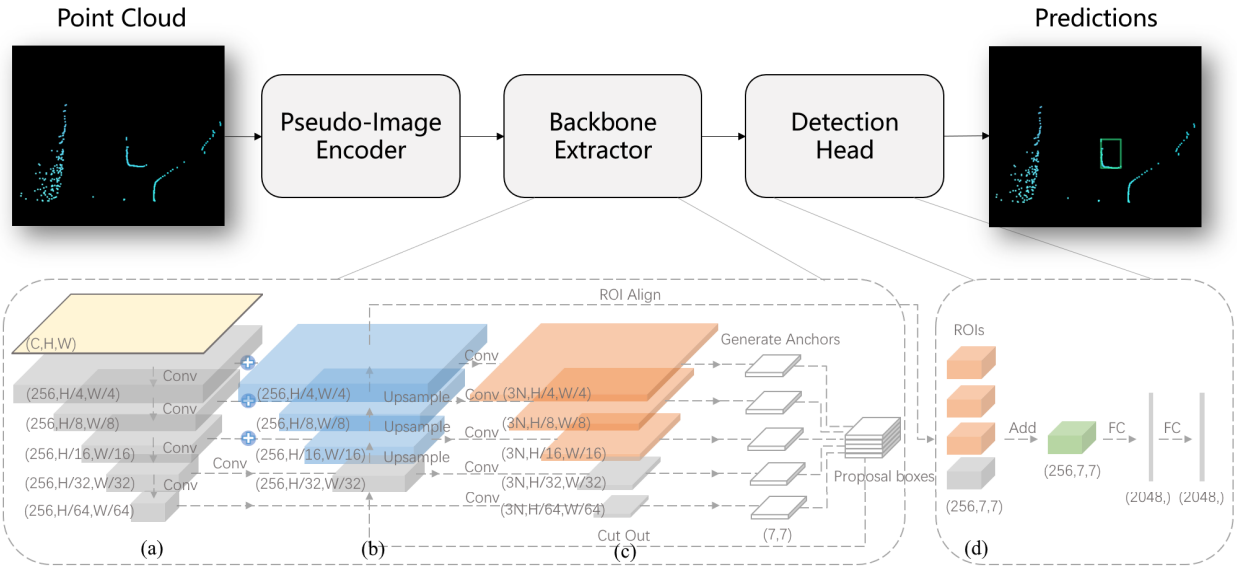


Fig. 7. Cascade Pyramid RCNN Network Architecture and Pipeline. Pseudo-image encoder first converts point cloud into a 3-dim tensor. Then backbone network extracts a feature map pyramid by convolution (a) and reversely constructs an upsampled pyramid (b) where on each level it is added by the corresponding feature map from (a). And the third pyramid (c) composes of feature maps from the previous two, with (b) passes through conv layer on each level, providing four levels of feature maps, and the last feature map is from (a) through another conv layer. Anchor boxes are generated within each layer in (c). Detection head collects valid proposal boxes to cut out and align RoI feature maps as (d). Orange tensors are the conv-ed sums of feature maps, blue tensors stand for upsampled sums of feature maps. The green tensor is the last feature map just before flattened into vectors. Three extra fully connected layers (FC) following the last vector estimate classification, location and orientation respectively, which are not shown here. Best viewed in color.

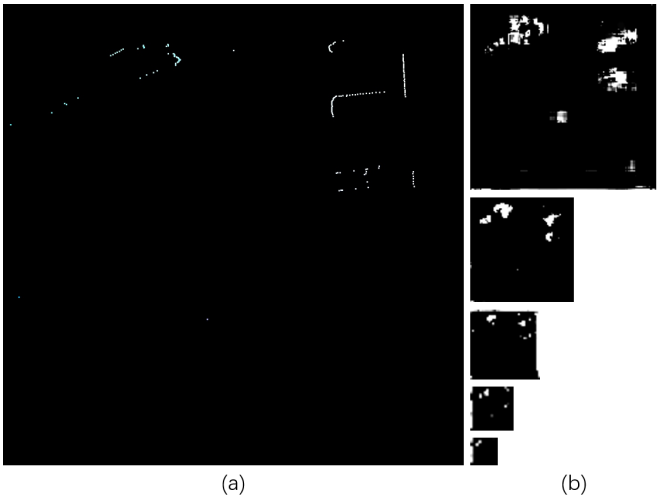


Fig. 8. Cascade Pyramid RCNN feature maps visualized. (a) original input with one annotated car at the top right. (b) feature maps of five scale levels from output of the backbone. As shown above, even though the input point clouds are basically sparse and hollow, the feature maps it learned are densified spontaneously. The network has learned to infer the densified point cloud clusters out of the original corner-and-edges.

coordinates on a unit circle.

$$\begin{aligned} \Delta\theta_x &= \lambda_{radius}\left(t_x - \sin\widehat{\theta}\right) \\ \Delta\theta_y &= \lambda_{radius}\left(t_y - \cos\widehat{\theta}\right) \end{aligned} \tag{1}$$

where λ_{radius} is the scaling radius of the unit circle in angle vector's Cartesian coordinates, $\Delta\theta_x$ and $\Delta\theta_y$ are the vector's projection on x axis and on y axis of a heading error. t_x and t_y are the vector's projection on x axis and on y axis of predicted heading outputs while $\widehat{\theta}$ stands for target heading in radian.

5) Implementation Details:

a) Data Augmentation: To enrich the diversity and further exploit the gathered data, we conduct several data augmentations: *Pseudo-image normalization*: The pseudo images are then subtracted by these means along each channel and divided by the standard deviations. and converted to BGR255 format for better compatibility with caffe pre-trained weights.

Object Noise: The noise factors are drawn from uniform distributions.

Global Rotation and Scaling to the whole point cloud and all ground-truth boxes: Each ground truth box and the point cloud are randomly transformed.

b) Network Settings: The network uses an x-y grid of 0.055 m, making the resolution of input pseudo image 600×600. The x, y range is $[(-3.33, 30), (-33.33, 33.33)]$ meters respectively.

Anchor generation and matching strategy are as follow: Each anchor is generated by width, length, center point, and orientation. The anchors in each feature map pyramid layer have sizes of (4,8,16,32,64) in pixels, strides of (8,16,32,64), and aspect ratios of (0.5,1.0,2.0). Re-sampling uses positive and negative ratios of 1:1 in RPN and 1:3 in RoI head. Anchors are matched to ground truth using the axis-aligned Intersection over Union (IoU) with gt's minimum bounding squares. A positive match is either the highest with a ground truth or above the positive matching threshold, while a negative match is below the negative threshold. Anchors that lie between are ignored in the loss. The threshold is 0.95/0.3 in RPN stage,

0.5/0.5 in RoI head. At inference time we apply axis aligned non-maximum suppression (NMS) with an overlap threshold of 0.3 IOU in RPN and 0.7 in RoI head. This provides similar performance compared with rotational NMS but is much faster.

The proposed network is trained using stochastic gradient descent (SGD) with momentum. The optimizer runs on 2 GTX 1080 Ti GPU with a total of 8 frames per minibatch. Models are trained for 100 epochs at maximum. The initial learning rate is 0.0005, with an exponential decay factor of 0.8 and a decay every 15 epochs. A weight decay regularization of 0.0005 and a momentum beta value of 0.9 are used. Training the car detection network on 2 GTX 1080 Ti GPU takes ∼24 hrs.

c) Loss Function: 1. Mean Square Error is used in angle regression after the vectorized angle encoded. Only positive samples are taken into calculation and the rest are ignored.

$$L_{ang} = \lambda_{ang} \sum_{b \in (x,y,\omega,l)} \left(\Delta\theta_x^2 + \Delta\theta_y^2 \right) \tag{2}$$

where λ_{ang} is the scaling factor of heading angle loss, x, y, ω, l are variables of center coordinates, width and length of a box b.

2. Focal loss [49] is proved to be able to ease the unbalanced sample problem in classification, both in sample quantity and recognition difficulty. Since the number of anchors and proposals greatly exceeds the possible number of targets, the foreground samples which are matched with ground truths are extremely outnumbered by background samples. That usually means insufficient learning for too little contribution can be made by valid samples in loss calculation that guides the network. By using focal loss, classification in these circumstances will be partly balanced and thus improved.

$$L_{cls} = -\alpha_{cls} \left(1 - p_{cls}\right)^{\gamma} \log\left(p_{cls}\right) \tag{3}$$

In which α_{cls} and γ are the scaling factors of classification loss, p_{cls} is the probability score of a certain category.

3. Smooth L1 loss [4] is introduced to prevent large values in box regressions using linear error outside $[-1, 1]$ while still keep a quadratic loss near the optimum to stay sensitive.

$$\begin{aligned}
\Delta x^{gt} &= \frac{x^{gt} - x^a}{\omega^a}, \quad \Delta y^{gt} = \frac{y^{gt} - y^a}{l^a} \\
\Delta \omega^{gt} &= \log \frac{\omega^{gt}}{\omega^a}, \quad \Delta l^{gt} = \log \frac{l^{gt}}{l^a} \\
\Delta x &= \frac{x - x^a}{\omega^a}, \quad \Delta y = \frac{y - y^a}{l^a} \\
\Delta \omega &= \log \frac{\omega}{\omega^a}, \quad \Delta l = \log \frac{l}{l^a} \qquad (4) \\
L_{loc} &= \sum_{b \in (x,y,\omega,l)} SmoothL1\left(\Delta b\right) \qquad (5)
\end{aligned}$$

Here, b, b^{gt} and b^a denote a box b's variables from prediction, ground truth and anchor respectively.

4.Total training loss is calculated by summing up the loss items above.

$$L_{total} = L_{cls} + L_{loc} + L_{ang} \tag{6}$$

6) Major Differences: First, unlike most architectures that can not handle bounding boxes with orientations, angle losses enable our method to regress arbitrary orientations. And a new kind of angle regression for rotated bounding boxes was introduced by using vectorized coordinates encodings. Second, fusions of different-scaled feature maps are repeatedly preformed along the model pipeline. As a contrast, a Faster-RCNN makes predictions only on the last layer of all feature maps and thus lacks the ability to integrate information from scale variant scenarios. Third, the classification loss function focusing on sample imbalance is used. Fourth, to match the proposed boxes in the first stage with ground truth boxes more efficiently, instead of directly matching two potential box pairs by the highest IoU, we preformed an axis-aligned IoU matching by calculating IoUs between a ground truth box's minimum bounding square and the proposed boxes.

B. Hybrid Resnet Lite

Since the 2D-LiDAR ranging data only contain a few hundred points, conversion to other forms may hurt the speeding up of the model. Besides, the pseudo image based method consumes too much computational resources, which makes it impossible to deploy on lightweight embedded platforms. Based on these motivation, we propose our second deep learning based method which can directly process data from 2D-LiDAR in form of points rather than pseudo images. The proposed method is composed of two modules. The first module is used to generate points region proposals, and the second module is a vehicle detector that uses the proposed point cluster to achieve classification and regression goals. *Hybrid* means our pipeline consists of both non-learning based and learning based modules, with 1D resnet kernels and Lite standing for its super light weight. The main architecture and pipeline are presented in Fig. 9.

1) Region Proposal Module: Different from the Region Proposal Network (RPN) that originates from [5] which generates anchor boxes over feature maps to predict multiple region proposals, we implement it using a more traditional algorithm named DBSCAN [50] to realize the same region proposal objective, which takes a frame of point cloud as input and outputs a set of point clusters as object proposals. The basic idea of the unsupervised DBSCAN algorithm (Density Based Spatial Clustering of Applications with Noise) is to divide points into different clusters based on two thresholds, one for the number of neighbors, MinPts, and the other for the distance measure,radius ϵ. The produced clusters of DBSCAN consist of core points and non-core points. The core point means that there are at least MinPts points within distance threshold ϵ, while the non-core point signifies that this point is directly reachable from a core point within distance threshold ϵ. Points unreachable from any core points are outliers. According to parameters tuning, the MinPts is set to 10 and the ϵ is 0.6 meter.

Considering the fact that most ranging points which belong to one target share sequential indices along the scanning direction, to simplify the region of interest's representation from storing all the points inside, we use 3 parameters

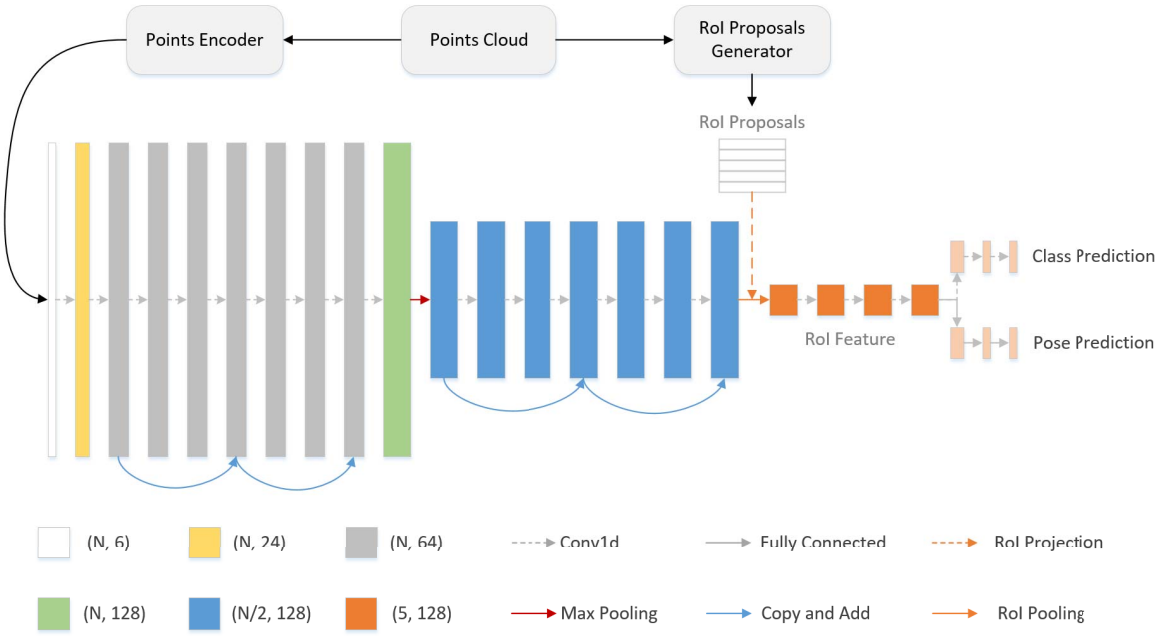


Fig. 9. Hybrid Resnet Lite Network Architecture and Pipeline. The network first extracts a global feature map for each point cloud with the re-coded points generated from Points Encoder. Then, the network cuts out the corresponding local feature map for each RoI based on encoded proposal vectors. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector. For each RoI, the network has two sibling outputs, the class prediction and the pose regression.

$[Idx_S, Idx_E, Num]$ to encode one region of interest. The Idx_S is region proposal's start index, the Idx_E is corresponding to the region proposal's end index and the Num means total points number of this point cloud input. In this way, the region of interest's data size can be greatly compressed.

2) Vehicle Detection Module: We design a convolution neural network named *Hybrid Resnet Lite* to realize the vehicle detection objective. The feature extraction layers are designed with the reference to ResNet [6], while convolution kernels are changed to 1 dimensional ones. To share computation between local and global feature maps, the feature extraction layers compute the global feature maps for all input points in one frame, then the network cuts out the corresponding local feature map for each region proposal based on encoded proposal vectors. Features of a proposal are extracted by adaptively maximum pooling the cut-out of the global feature map into a fixed-size output. The above process is inspired by the RoI pooling proposed in [4].

For each region proposal, this vehicle detector has two sibling outputs. The first output is the RoI's probability p to the vehicle category, and correspondingly $1 - p$ is probability to background. The second output encodes the target's pose estimation, i.e. localization offsets and heading angle, $[t_x, t_y, \theta]$.

3) Implementation Details:

a) Point Data encoding: The original ranging points collected by 2D-LiDAR are represented in Cartesian coordinates, each ranging point only encodes 2 numbers, which makes it difficult for the network to extract general and effective features among vehicles in different positions. Thus, we manage to reduce that difficulty by encoding each point with 6 parameters, $[x, y, \rho, \Delta x, \Delta y, \Delta\rho]$. x, y are the original ranging points coordinates, and ρ is the distance from the ranging point to the coordinate origin. For points belonged to a specific proposal region, Δx, Δy are the relative coordinates to the proposal region's center, and $\Delta\rho$ is the corresponding relative distance to the proposal region's center. For points that do not belong to any proposal region, Δx, Δy, $\Delta\rho$ are all set to zeros.

b) Multi-task Loss: To simultaneously estimate the vehicle's position and heading, each RoI during training is labeled with a ground-truth class, position offsets and a heading angle. We use a multi-task loss L on each labeled RoI's back-propagation to jointly train up classification and pose regression:

$$L = \lambda_1 L_{cls} + \lambda_2 L_{loc} \tag{7}$$

t in which L_{cls} is binary cross entropy loss and L_{loc} is mean squared error loss. During the training phase, if an RoI is labeled with negative ground truth class, the L_{loc} loss will be ignored. The heading angle was encode by two correlated values $cos(\theta), sin(\theta)$ to enhance the angle range regression robustness. The two hyper-parameters λ_1 and λ_2 in Eq. 7 maintain the balance between two task losses.

V. Experiments and Evaluations

Our experiments over the dataset are carried out with the following evaluation metrics: the predictions are evaluated by their center points and heading directions, the different criterion of distance to targets center points and angular errors from targets heading directions are tested respectively. A prediction is considered accurate only when it's center point offset and heading direction error are both below certain thresholds. Result summary is shown in Table II.

TABLE II
SUMMARY OF EXPERIMENT AND ABLATION STUDIES

Model	Pseudo Image	Feature Extractor	Detection Head	Heading	Proposal	AP* (%)	Speed (ms)
Cascade Pyramid RCNN	Encoded	Pyramid	Pyramid head	Vectorized	Square	**88.2 (89.8)**	48.2
Pyramid RCNN	Encoded	Pyramid	Non-Pyramid head	Vectorized	Square	75.5	46.4
Pyramid RCNN	Encoded	Non-Pyramid	Pyramid head	Vectorized	Square	-[1]	-
Pyramid CNN	Encoded	Pyramid	-[2]	Vectorized	Square	69.5	36.3
Cascade Pyramid RCNN	Encoded	Pyramid	Pyramid head	Vectorized	Direct	76.6	48.1
Cascade Pyramid RCNN	Encoded	Pyramid	Pyramid head	Radian	Square	65.4	48.2
Cascade Pyramid RCNN	Whitened	Pyramid	Pyramid head	Vectorized	Square	82.5	48.2
Hybrid Resnet Lite	-	-	-	-	-	76.3 (78.9)	**9.1**
Faster RCNN[4] [5]	Encoded	Non-Pyramid	Non-Pyramid head	Radian	Direct	-[3]	-
Faster RCNN(with angle loss)[5]	Encoded	Pyramid	Non-Pyramid head	Radian	Direct	44.7 (68.8)	45.3

* Evaluated under 30cm positioning error and 15° heading error criterion. Values in bracket follow the criteria of positioning error only
[1,3] Without the 1st pyramid, network does not converge, thus unable to complete the task.
[2] The detection head is removed entirely.
[4,5] To fairly compare, angle loss is added in the same way as Cascade Pyramid RCNN.

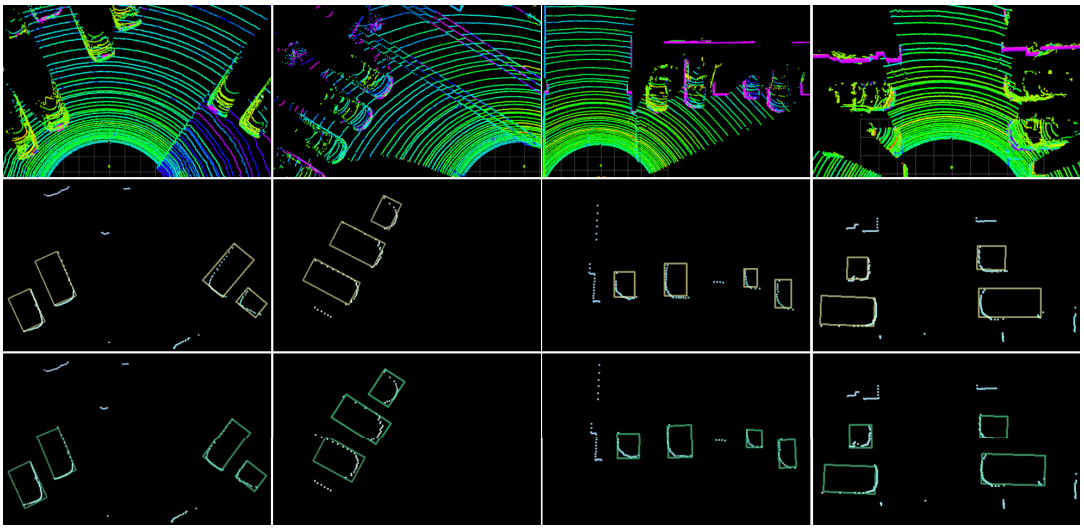


Fig. 10. Detection results demonstrated in bird's-eye view. Upper row: 3d point clouds for visualization purpose; Middle row: Ground truth boxes of the vehicles in 2D point clouds in yellow; Lower row: Predicted boxes of the vehicles in 2D point clouds in green. Best viewed in color.

A. Quantitative Analysis

The Cascade Pyramid RCNN achieves around an Average Precision (AP) of 88.19% with a positive criterion of within 30 cm in radius and a heading angle error less than 15°. Grid resolution is 600 × 600 (Later experiments follow this same criteria if not otherwise specified). Comparisons with different criterion thresholds (distance radius and heading deviation) prove the robustness of the network. With various distance and heading criterion, the amplitude of variation is largely steady.

As for the inference time, Cascade Pyramid RCNN takes about 48 ms per frame, i.e., above 20Hz fresh rate, which is pretty competent for real-time detection. Note that the network is not factorized or squeezed [51], [52] [53] specifically for faster speed, thus it still remains much potential in exploiting that.

From the experiment results, intuitively, as the projected grid's resolution goes higher, the average precision is giving a significant rise. This is most likely because a finer grid preserves more accurate location information from raw point clouds as fewer points would be encoded into same grids. However, when the resolution reaches an inflection point, the average precision turns into negative correlation with resolution changes, probably due to the increased sparsity that can no longer be fully compensated. Therefore, a sweet spot of most suitable grid resolution can be concluded according to the network structure used.

Different detecting ranges result shows that 2D LiDAR detection performance gradually increases as point clouds become closer to the sensor's view point. The reason could be the increasing density of information. We tested square ranges of $(16.67^2, 15^2, 10^2, 7^2)m$ in front of the ego-vehicle. The results re-confirm that it is more suitable for detection tasks in relatively near distance such as blind spot detection and parking lots scenarios.

Different encoding results are also without a surprise. The encoding method that consists of the coordinates x, y, and distance to observing point as the three channel values (x,y,d)

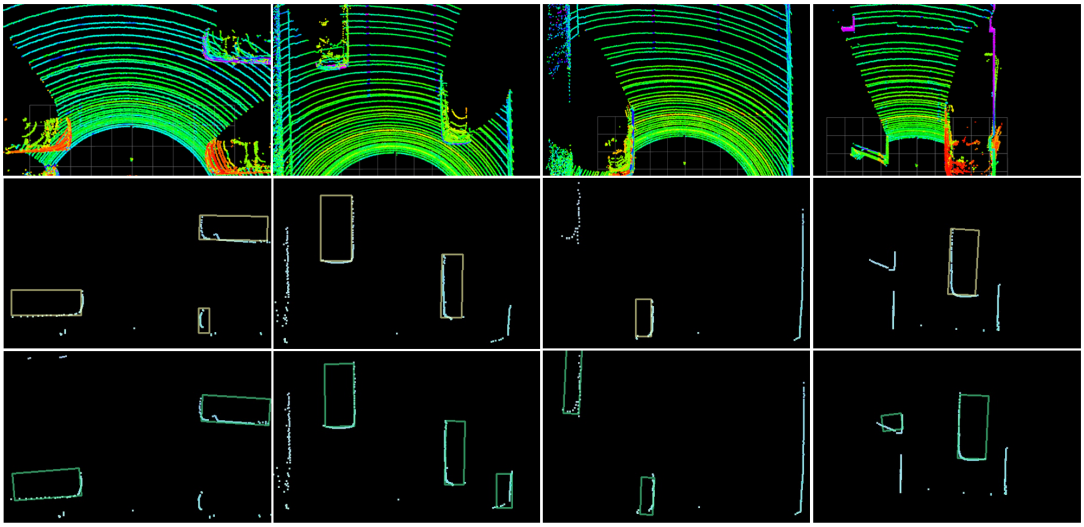


Fig. 11. Detection failure cases demonstrated in bird's-eye view. (a) Missing detection: a vehicle at the lower-right corner is missed. (b) False positive: road side fence is detected as a vehicle. (c) False positive: road side bushes is detected as a vehicle. (d) False positive: a corner is detected as a vehicle.

TABLE III
COMPARISONS WITH OTHER METHODS

Model	AP* (%)	AP with orientation (%)
Cascade Pyramid RCNN	**89.8**	**88.2**
Hybrid Resnet Lite	78.9	76.3
Faster RCNN(with angle loss)	68.8	44.7
Faster RCNN	89.4	-
SSD	81.6	-
Retinanet	90.9	-

*Evaluated under 30cm positioning error criteria. Any angle error of predicted boxes is not taking into account since all original baseline models like Faster RCNN, SSD and Retinanet are not capable of regressing box orientations.

TABLE IV
CASCADE PYRAMID RCNN AND HYBRID RESNET LITE COMPARISON

		ΔAP*(%)			
		$0.30m\_15°$	$0.30m\_25°$	$0.45m\_15°$	$0.45m\_25°$
$X_c(m)$	$0 \sim 3$	11.33	14.37	2.62	5.32
	$3 \sim 6$	8.04	6.18	0.43	0.69
	$6 \sim 9$	6.28	4.22	2.74	0.95
	$9 \sim \infty$	20.0	20.1	15.0	15.0
$\theta(°)$	$0 \sim 45$	6.22	7.52	2.46	3.76
	$45 \sim 90$	16.59	14.57	10.56	7.97
	$90 \sim 135$	5.54	6.14	4.13	4.63
	$135 \sim 180$	-0.48	0.42	0.30	0.51
$R(m)$	$0 \sim 5$	4.95	8.35	2.34	2.44
	$5 \sim 10$	9.94	10.9	4.88	3.0
	$10 \sim \infty$	18.78	18.78	11.74	11.74

*Cascade Pyramid RCNN and Hybrid ResNet Lite comparison from three aspects: target distance r, its projection x_c on x axis, and target relative angle $\theta = \gamma - \alpha$, where $\gamma = tan(x_c, y_c)$, α is target heading angle with respect to x axis. $\Delta AP(\%)$ equals to *Cascade Pyramid RCNN* AP performance minus *Hybrid Resnet Lite* AP performance.

of a pseudo image tensor, exceeds the performance from a uniformed whitened encoding method by some margin. Encoded grids seem to preserve more information from the original data. So an effective encoding could do some contribution to the final performance, although not much can be exploited from 2d point cloud.

The Hybrid Resnet Lite also achieves much higher performance than Faster RCNN when evaluate with a positive criterion of within 30 cm in radius and a heading angle error less than 15°, the average performance gain of AP is 31.6%. Moreover, the inference speed of Hybrid Resnet Lite is also much faster than the Faster RCNN, with an average gain of 400%, from 45.3 $ms(22fps)$ to 9.1$ms(110fps)$. Though the overall performance can be less accurate than the Cascade Pyramid RCNN, the significant reduction in time consumption could be very useful.

B. Qualitative Analysis

We present our detection results as shown in Fig. 10, with oriented 2D bounding boxes. The light blue points represent the encoded pixels from point clouds, yellow boxes with pointing lines are hand labeled ground truth while green boxes stand for model predictions. The predictions for surrounding vehicles seem largely accurate. The networks are well capable of distinguish vehicle contours from noise, background and other categories of objects. On the top of that, the regression of a distinguished targets' position and orientation is only with minimal errors.

Although in some minor cases, as in Fig. 11 prediction failures do occur sometimes in forms of false positives on obscure obstacles, false negative on difficult samples. In Fig. 11 (a), cars with too few points or limited view can be sometimes missed by the model, probably due to occlusions or unsuitable viewpoint. While in (b), (c) and (d), roadside bushes and corners may resemble the shape of a car and thus occasionally produce confusing outputs.

Though in most cases, targets are correctly recognized and located with heading angles, it is uneasy to determine the actual feature that the network has learned. To get a hint of what was learned, we extract the feature maps as in Fig. 8 and

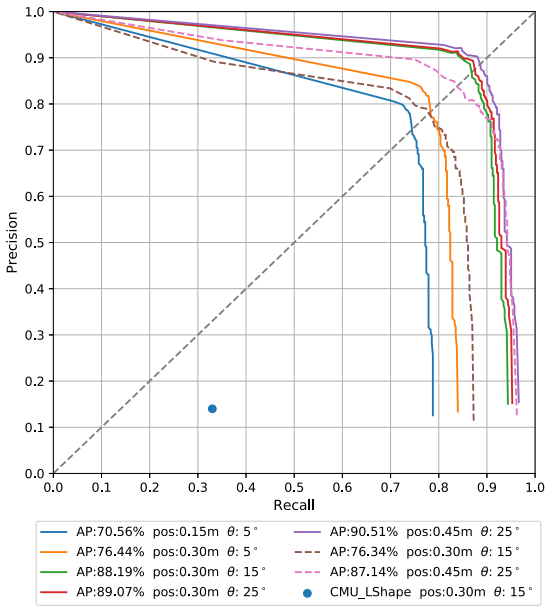


Fig. 12. Precision-Recall curves for different positioning and heading error criterion. Performance is given in average precision (AP). Solid lines for Cascade Pyramid RCNN and dash lines for Hybrid Resnet Lite.

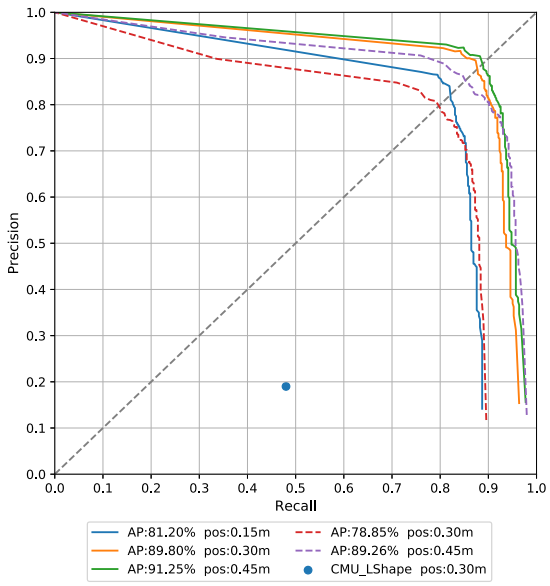


Fig. 13. Precision-Recall curves for different positioning error criterion only, without considering heading. Performance is given in average precision (AP). Solid lines for Cascade Pyramid RCNN and dash lines for Hybrid Resnet Lite.

the feature maps show that the network has learned to infer the densified point cloud clusters out of the original corner-and-edges. These inferred maps are reasonable and understandable because they conforms with the loss constraints we mentioned in section IV, including positioning constraints, longitudinal and widthwise size constraints, and orientation constraint.

C. Methods Comparisons

As presented in Fig. 12, 13, 14, and 15, the Cascade Pyramid RCNN outperforms current existing methods with a large margin, both when compared with the non-learning based method [22] and learning-based one [5] as in Table III. The Hybrid Resnet Lite method also achieves competitive result considering it is an ultra light weighted model at just 5MB and can inference at a much higher speed of ∼9ms per frame, while it still holds state of the art target positioning capability. Since both proposed methods are capable of performing accurate detection tasks, they can be applied for different uses. Generally, Cascade Pyramid RCNN is more suitable for scenarios that require higher detection accuracy and more robustness. While thanks for the lightweight design, Hybrid ResNet Lite can be deployed on embedded systems with limited computing resources.

D. Methods Discussion

To further combine the best of each method, more detailed discussion about which method should be taken in different scenarios is listed in Table IV. We evaluate and compare the two method from multiple aspects, including target distance r, its projection x_c on x axis, and target relative angle $\theta = \gamma - \alpha$, where $\gamma = tan(x_c, y_c)$, α is target heading angle with respect to x axis. The chart shows that the two methods provide similar performance when detecting targets from mid and remote distance. For targets of very close or very remote

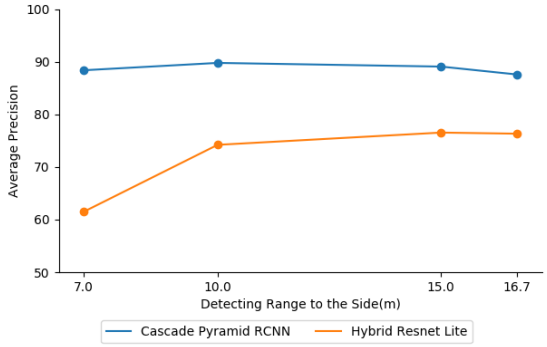


Fig. 14. Impact of different detecting range on average precision. Evaluating with detecting areas of different sized squares in front of the car body, i.e. $16.7^2; 15^2; 10^2; 7^2 m$.

distance, Cascade Pyramid RCNN produces substantially more accurate output. But if the application scenario allows for a little more relaxed evaluate criteria, the two can still achieve very similar performance, which also means that we could switch to Hybrid ResNet to save computation resources. In other words, a system can be further optimized by switching the two methods in different distance ranges, using Hybrid Resnet Lite for mid and remote range detection for better efficiency, and using Cascade Pyramid RCNN for very close and very remote areas or certain focused areas that require both better location and heading angle accuracy.

VI. Ablation Study

To figure out the key factors that enable the performance of our proposed Cascade Pyramid RCNN, we conduct ablation studies shown in Table. II and explain why they make sense for our design choices.

c) Cascade Pyramid Structure: Due to the limitation of 2D field of view and occlusions, point clusters that belong to the target objects like cars may be very sparse or different

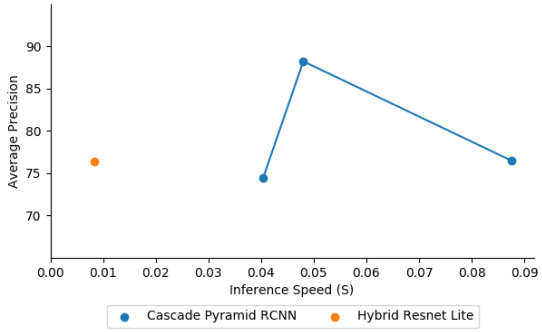


Fig. 15. Inference time vs. average precision. Different operating points of Cascade Pyramid RCNN are achieved by using grid sizes of $3.33^2; 5.55^2; 16.66^2 cm$ which are resolutions of $200^2; 600^2; 1000^2$ respectively.

in sizes and scales. The introduction of the cascade pyramid is just about easing these difficulties. To assess the real impact, we implemented Cascade Pyramid RCNN with different feature map pyramids removed. Performance in average precision shows that cascade pyramid does gain a large margin over the rest models. Notably, if the first feature map pyramid is removed, the model does not even converge. Our hypothesis is that combining different spatial information in different levels and at different stages of a network's pipeline plays a crucial role in detecting targets with features that are hard to capture due to various sparsity, scales and field of view limitations.

d) Vectorized Angle Representaion: To determine the effect that vectorized angle representation brings about, another model of Cascade Pyramid RCNN is trained with exactly the same hyper-parameters and structures, with only different angle value encode and decode rules. Instead of encoding angles into the vectorized coordinates, the compared model uses the same two values to predict one target angle in radian and then average them. Result shows without the vectorized angle representation, performance drops dramatically, since vehicle heading directions can be rather arbitrary, it comes with more frequent singularity in certain circumstances to overcome.

e) Two-Stage Region Proposal: Although the two-stage pipeline of Cascade Pyramid RCNN is relatively slow compared with one stage methods like YOLO [54] or SSD [55], the two-stage methods seem to often outperform the single stage methods, as it does in our experiment where we remove the RoI detection head entirely and receives poorer performance. It is mainly because the offsets are regressed twice hierarchically, which gives finer estimation one based on another.

f) Square Proposal Matching: Lastly, to cut out oriented proposal feature maps during training, we implemented the axis-aligned cut-out for each proposal box, for it is faster and simpler than the oriented process and still be accurate enough. To keep coherence with that, during the box matching process, each oriented box is treated as its axis-aligned minimum bounding square. Ablation experiment shows that without square matching in proposing stage, the axis-aligned matching performance is reduced. Because proposed feature maps are possibly cut out in a way that they do not have a full perceptive field of targets.

VII. Conclusion

In this work, we revisit the LiDAR based approach for vehicle detection with a less expensive 2D LiDAR by utilizing modern deep learning approaches. We propose our first learning based method with the input of pseudo-images, named Cascade Pyramid Region Proposal Convolution Neural Network (Cascade Pyramid RCNN). Our second approach is a hybrid learning method with the input of sparse points, named Hybrid Resnet Lite. Our experimental results demonstrate that the Cascade Pyramid RCNN outperforms state-of-the-art methods in accuracy while the proposed Hybrid Resnet Lite provides superior performance of the speed and lightweight model by hybridizing learning based and non-learning based modules.

Our work successfully addresses the challenges of 3D LiDAR based and vision based methods, and offers an alternative for robust and efficient vehicle detection method based on less expensive 2D LiDAR. As few previous works conclude an efficient and robust vehicle detection solution with 2D LiDAR, our research fills in this gap and illustrates that even with limited sensing source from a 2D LiDAR, detecting obstacles like vehicles efficiently and robustly is still achievable.

Considering future work, networks can be specifically squeezed or factorized like in [51], [52] [53] to slim the kernel and speed up without major performance loss. Also, sparse convolution [56] can be introduced to further minimize computation and reduce inference time. Such enhancements could help make better trade off between models performance and speed. Since this research focuses on demonstrating and improving the feasibility of deep learning architectures in detecting surrounding cars from a single frame and a single 2d lidar, supplementary information or techniques would also be helpful in future works.

Acknowledgment

The authors would like to thank Junqiao Zhao for his autonomous driving platform TiEV.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[2] P. Perconti and A. Plebe, "Deep learning and cognitive science," *Cognition*, vol. 203, Oct. 2020, Art. no. 104365.
[3] Y. Liu *et al.*, "Motor-Imagery-Based teleoperation of a dual-arm robot performing manipulation tasks," *IEEE Trans. Cognit. Develop. Syst.*, vol. 11, no. 3, pp. 414–424, Sep. 2019.
[4] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2961–2969.
[8] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4151–4160.
[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[10] G. Chen, H. Cao, J. Conradt, H. Tang, F. Rohrbein, and A. Knoll, "Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 34–49, Jul. 2020.

[11] G. Chen, L. Hong, J. Dong, P. Liu, J. Conradt, and A. Knoll, "EDDD: Event-based drowsiness driving detection through facial motion analysis with neuromorphic vision sensor," *IEEE Sensors J.*, vol. 20, no. 11, pp. 6170–6181, Jun. 2020.

[12] G. Chen *et al.*, "A survey of the four pillars for small object detection: Multi-scale representation, contextual information, super-resolution, and region proposal," *IEEE Trans. Syst., Man Cybern., Syst.*, 2020, doi: 10.1109/TSMC.2020.3005231.

[13] K. O. Arras, O. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D range data," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 3402–3407.

[14] L. Spinello and R. Siegwart, "Human detection using multimodal and multidimensional features," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 3264–3269.

[15] C. Weinrich, T. Wengefeld, C. Schroeter, and H.-M. Gross, "People detection and distinction of their walking aids in 2D laser range data based on generic distance-invariant features," in *Proc. 23rd IEEE Int. Symp. Robot Human Interact. Commun.*, Aug. 2014, pp. 767–773.

[16] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2D laser scanners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 726–733.

[17] J. Schlichenmaier, F. Roos, M. Kunert, and C. Waldschmidt, "Adaptive clustering for contour estimation of vehicles for high-resolution radar," in *IEEE MTT-S Int. Microw. Symp. Dig.*, May 2016, pp. 1–4.

[18] J. Schlichenmaier, N. Selvaraj, M. Stolz, and C. Waldschmidt, "Template matching for radar-based orientation and position estimation in automotive scenarios," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Mar. 2017, pp. 95–98.

[19] X. Shen, S. Pendleton, and M. H. Ang, "Efficient L-shape fitting of laser scanner data for vehicle pose estimation," in *Proc. IEEE 7th Int. Conf. Cybern. Intell. Syst. (CIS) IEEE Conf. Robot., Autom. Mechatronics (RAM)*, Jul. 2015, pp. 173–178.

[20] R. MacLachlan and C. Mertz, "Tracking of moving objects from a moving vehicle using a scanning laser rangefinder," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2006, pp. 301–306.

[21] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Auto. Robots*, vol. 26, nos. 2–3, pp. 123–139, Apr. 2009.

[22] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient L-shape fitting for vehicle detection using laser scanners," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 54–59.

[23] S. Qu *et al.*, "An efficient L-Shape fitting method for vehicle pose detection with 2D LiDAR," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2018, pp. 1159–1164.

[24] L. Beyer, A. Hermans, and B. Leibe, "DROW: Real-time deep learning-based wheelchair detection in 2-D range data," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 585–592, Apr. 2017.

[25] L. Beyer, A. Hermans, T. Linder, K. O. Arras, and B. Leibe, "Deep person detection in 2D range data," 2018, *arXiv:1804.02463*. [Online]. Available: http://arxiv.org/abs/1804.02463

[26] Á. M. Guerrero-Higueras *et al.*, "Tracking people in a mobile robot from 2D LIDAR scans using full convolutional neural networks for security in cluttered environments," *Frontiers Neurorobotics*, vol. 12, p. 85, Jan. 2019.

[27] W. Lin *et al.*, "Group reidentification with multigrained matching and integration," *IEEE Trans. Cybern.*, early access, Jun. 11, 2019. [Online]. Available: http://europepmc.org/abstract/med/31199281

[28] W. Lin *et al.*, "Learning correspondence structures for person re-identification," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2438–2453, May 2017.

[29] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.

[30] 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 *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1355–1361.

[31] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1513–1518.

[32] S. Wirges, T. Fischer, J. Balado Frias, and C. Stiller, "Object detection and classification in occupancy grid maps using deep convolutional networks," 2018, *arXiv:1805.08689*. [Online]. Available: http://arxiv.org/abs/1805.08689

[33] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "SBNet: Sparse blocks network for fast inference," 2018, *arXiv:1801.02108*. [Online]. Available: http://arxiv.org/abs/1801.02108

[34] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Proc. Conf. Robot Learn.*, Oct. 2018, pp. 146–155.

[35] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7652–7660.

[36] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," 2016, *arXiv:1608.07916*. [Online]. Available: http://arxiv.org/abs/1608.07916

[37] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1907–1915.

[38] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.

[39] K. Shin, Y. Paul Kwon, and M. Tomizuka, "RoarNet: A robust 3D object detection based on RegiOn approximation refinement," 2018, *arXiv:1811.03818*. [Online]. Available: http://arxiv.org/abs/1811.03818

[40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," 2016, *arXiv:1612.00593*. [Online]. Available: http://arxiv.org/abs/1612.00593

[41] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*. [Online]. Available: http://arxiv.org/abs/1706.02413

[42] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[43] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," 2018, *arXiv:1812.04244*. [Online]. Available: http://arxiv.org/abs/1812.04244

[44] Z. Wang and K. Jia, "Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," 2019, *arXiv:1903.01864*. [Online]. Available: http://arxiv.org/abs/1903.01864

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

[46] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," 2018, *arXiv:1812.05784*. [Online]. Available: http://arxiv.org/abs/1812.05784

[47] J. Zhao *et al.*, "TiEV: The tongji intelligent electric vehicle in the intelligent vehicle future challenge of China," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1303–1309.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[49] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[50] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, vol. 96, no. 34, pp. 226–231.

[51] N. I. Forrest, H. Song, W. Matthew, A. Khalid, and J. W. Dally, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," in *Proc. Conf. (ICLR)*, 2017, pp. 207–212.

[52] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[53] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[54] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[55] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[56] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," 2017, *arXiv:1706.01307*. [Online]. Available: http://arxiv.org/abs/1706.01307