Joint 3D Proposal Generation and Object Detection from View Aggregation

Jason Ku
jason.ku@uwaterloo.ca

Melissa Mozifian
melissa.mozifian@uwaterloo.ca

Jungwook Lee
j343lee@uwaterloo.ca

Ali Harakeh
www.aharakeh.com

Steven Waslander
stevenw@uwaterloo.ca

Department of Mechanical and Mechatronics Engineering, WAVE Lab
University of Waterloo, Waterloo, ON, Canada

Abstract

*We present AVOD, an **A**ggregate **V**iew **O**bject **D**etection network for autonomous driving scenarios. The proposed neural network architecture uses LIDAR point clouds and RGB images to generate features that are shared by two subnetworks: a region proposal network (RPN) and a second stage detector network. The proposed RPN uses a novel architecture capable of performing multimodal feature fusion to generate reliable 3D object proposals for multiple object classes in road scenes. Using these proposals, the second stage detection network performs accurate oriented 3D bounding box regression and category classification to predict the extents, orientation, and classification of objects in 3D space. Our proposed architecture is shown to produce state of the art results on the KITTI 3D object detection benchmark [1] while running in real time with a low memory footprint, making it a suitable candidate for deployment on autonomous vehicles.*

1. Introduction

The remarkable progress made on the task of 2D object detection in recent years has not transferred well to the detection of objects in 3D. The gap between the two remains large on standard benchmarks such as the KITTI Object Detection Benchmark [1] where 2D *car* detectors such as RRC [2] have achieved over 90% Average Precision (AP), whereas the top scoring 3D *car* detector on the same data only achieves around 70% 3D AP and 84% bird's eye view (BEV) AP. The reason for such a gap stems from the difficulty induced by adding a third dimension to the estimation problem, the low resolution of 3D input data, and the deterioration of its quality as a function of distance.

Most current 3D object detection methods leverage multimodal input from cameras and 3D range sensors. Methods using LIDAR point clouds [3, 4, 5, 6], stereo depth maps [7], or RGBD sensor depth maps [8] in addition to images have been shown to outperform image only methods [9, 10, 11] on the 3D object detection task. Multimodal object detectors have been previously discussed in the literature [12, 13, 14, 15, 3, 7], but up to our knowledge, no previous work has exploited features from both point cloud and RGB data to generate region proposals. Ignoring either source discards important information that can be used for generating better region proposals, and in particular, leads to lower performance when targeting small object classes such as pedestrians and cyclists.

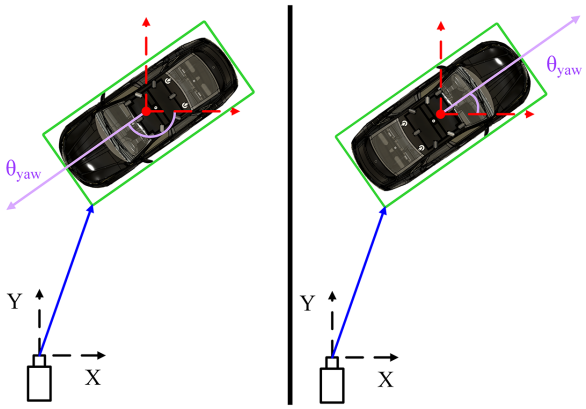

Figure 1: Parameters employed by metrics commonly used to characterize the performance of 3D object detectors. **Green:** The bounding box used to determine the IoU overlap in the computation of the *average precision*. **Blue:** The vector used to compute *the average error in distance to impact*. **Red:** The centroid of the bounding box used to calculate the *average error in estimating the centroid*. It can be seen that the parameter values, and therefore the performance metrics, are unchanged when the orientation (**purple**) is shifted by $\pm\pi$ radians.

An additional difficulty specific to the 3D object detec-

tion task is the creation of descriptive performance metrics to reliably capture detector performance. Previously proposed metrics[1, 10], such as the 3D and BEV AP, the average error in distance to impact, and the average error in estimating the centroid, all fail to fully capture the performance of 3D object detectors. These metrics primarily assess the localization and classification performance, but fail to penalize the error of the detectors on one crucial output of 3D object detection; the global orientation angle estimate. For autonomous driving datasets such as KITTI, the global orientation is provided in the form of the yaw angle, i.e. the object's heading. Average precision is currently the only metric used by the *KITTI 3D benchmark* that captures errors in orientation. The AP metric classifies any box with an IoU greater than a certain threshold with any ground truth box as a correct detection. Shifts in orientation that preserve an IoU greater than the required threshold are not penalized in the current KITTI evaluation process (Fig. 1).

In this paper, we aim to resolve these difficulties by proposing AVOD, an **A**ggregate **V**iew **O**bject **D**etection architecture for autonomous driving (Fig. 2). The proposed neural network architecture uses LIDAR point clouds and RGB images to generate features that are shared by two subnetworks: a region proposal network (RPN) and a second stage detector network. The RPN is based on a novel architecture that is capable of using features from multiple input modes for the proposal generation task, allowing for higher recall proposal generation on smaller objects in road scenes. Using these proposals, the second stage detection network performs accurate *oriented* 3D bounding box regression predicting the extents, orientation, and category classification of objects in 3D. We evaluate the proposed architecture on the tasks of proposal generation, 3D detection, and bird's eye view (**BEV**) detection on the challenging KITTI 3D object detection benchmark. At the time of this paper's submission, AVOD ranks **first** in BEV and **second** in 3D for cars, and **third** for pedestrians and cyclists on the KITTI 3D object detection benchmarks. Furthermore, the network is shown to perform in real time with limited memory requirements, making it a suitable candidate for deployment on autonomous vehicles. Finally, we show that current metrics used by the KITTI 3D object detection benchmark do not fully characterize the performance of 3D object detectors and extend the Average Orientation Similarity (AOS) measure [1] from KITTI's 2D object detection benchmark to capture the performance of 3D detectors on orientation estimation in the global coordinate frame. We refer to this measure as the *Average Heading Similarity* (AHS) and show that the proposed architecture outperforms the state of the art MV3D architecture [3] by a large margin when using the AHS alongside 3D AP for evaluation.

2. Related Work

Before the emergence of 3D Region Proposal Networks (RPNs) [16], 3D proposal generation algorithms typically used hand-crafted features to generate a small set of candidate boxes that retrieve most of the objects in 3D space. 3DOP [17] uses hand-crafted geometric features from stereo point clouds to score 3D sliding windows in an energy minimization framework. The top K scoring windows are selected as region proposals, which are then consumed by a modified Fast-RCNN [18] to generate the final 3D detections. Mono3D [9] uses the same framework, but instead exploits a ground plane prior and hand-crafted features from semantic segmentation outputs to generate 3D proposals from monocular images. We use a region proposal network that learns features from both BEV and image spaces to generate higher quality proposals in an efficient manner.

Single shot object detectors similar to YOLO [19] have also been proposed as RPN free architectures for the 3D object detection task. VeloFCN [4] projects a LIDAR point cloud to the front view, which is used as an input to a fully convolutional neural network to directly generate dense 3D bounding boxes. 3D-FCN [5] extends this concept by applying 3D convolutions on 3D voxel grids constructed from LIDAR point clouds to generate better 3D bounding boxes. Our two-stage architecture uses an RPN to retrieve most object instances in the road scene, providing better results when compared to both of these single shot methods.

Another alternative to 3D RPN-based architectures is 3D object detection from monocular images only. Deep MANTA [11] proposes a many-task vehicle analysis approach from monocular images that optimizes region proposal, detection, 2D box regression, part localization, part visibility, and 3D template prediction simultaneously. The architecture requires a database of 3D models corresponding to several types of vehicles, making the proposed approach hard to generalize to classes where such models do not exist. Deep3DBox [10] proposes to extend a 2D object detector [20] to 3D by exploiting the fact that the perspective projection of a 3D bounding box should fit tightly within its 2D detection window. However, these methods run over $8\times$ slower than our proposed method, and in Section 5, these methods are shown to perform poorly on the 3D detection task compared to methods that use point cloud data.

Recently, VoxelNet [21] was released as a single shot detector that leverages more powerful point-wise features extracted directly on a 3D voxel grid. However, even with sparse 3D convolution operations, the computational speed is still $3\times$ slower than our proposed architecture with only a slight increase in performance on the *pedestrian* and *cyclist* classes.

One final alternative for RPNs present in literature is us-

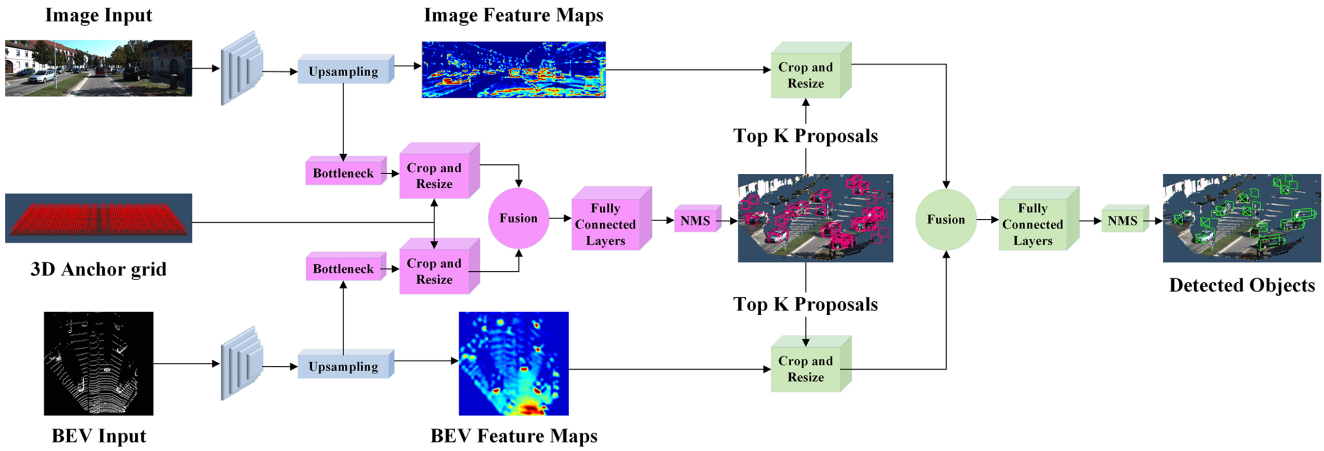

Figure 2: The proposed method's architectural diagram. The feature extractors are shown in **blue**, the region proposal network in **pink**, and the second stage detection network in **green**.

ing mature 2D object detectors for proposal generation in 2D, followed by amodal 3D extent regression. This trend started with [22] for indoor object detection, which inspired Frustum-based PointNets (F-PointNet) [23] to extend the feature extraction stage to rely on point-wise features of PointNet [24] instead of point histograms. While these methods work well for indoor scenes and brightly lit outdoor scenes, they are expected to perform poorly in more extreme outdoor scenarios. This is mainly due to sensor noise and severe lighting variations that may be present in autonomous driving scenarios. In Section 5 we show that AVOD is robust to noisy LIDAR data and lighting changes, as it was tested in snowy scenes and in extreme low light conditions. This is due to the fusion mechanism at the RPN stage, allowing reliable proposal generation even when LIDAR or image quality are degraded.

3D RPNs have previously been proposed in [8] for 3D object detection from RGBD images. However, up to our knowledge, MV3D [3] is the only architecture that proposed a 3D RPN targeted at autonomous driving scenarios. MV3D extends the image based RPN of Faster R-CNN [16] to 3D by corresponding every pixel in the *BEV* feature map to multiple prior 3D anchors. These anchors are then fed to the RPN to generate 3D proposals that are used to create view-specific feature crops from the BEV, front view of [4], and image view feature maps. A *deep fusion* scheme is used to combine information from these feature crops to produce the final detection output.

The Faster R-CNN RPN architecture is tailored for dense, high resolution image input, where objects usually occupy more than one pixel in the feature map. When considering sparse and low resolution input such as the BEV point cloud projection, this method is not guaranteed to have enough information from a single pixel to generate region proposals. As an example, an average pedestrian in the KITTI dataset occupies 0.8×0.6 meters in the BEV. This translates to an 8×6 pixel area in a BEV map with 0.1 meter resolution. When downsampled by convolutional feature extractors, instances from this class will occupy a fraction of a pixel in the resultant feature map. Even with higher resolution BEV maps, the sparsity of LIDAR data is not sufficient to extract informative features for these small sized classes. Our architecture aims to fuse features from the image and the BEV feature maps as inputs to the RPN, allowing the generation of high recall proposals for smaller classes. One final shortcoming of MV3D is its heading estimation mechanism that relies on extracting heading information from the regressed 8 corners of a 3D bounding box (Fig. 3). We show in Section 5 that this method cannot be used to extract unique orientation information from only the corners of the bounding box. Our architecture explicitly regresses a orientation vector that is used to correct the orientation estimate extracted from such box representations. This is shown to provide a 29.46% increase in AHS over MV3D on the *car* class of the KITTI dataset.

3. The AVOD Architecture

The proposed method, depicted in Fig. 2, uses feature extractors to generate feature maps from both the BEV map and the RGB image. Both feature maps are then used by the RPN to generate non-oriented region proposals, which are passed to the detection network for dimension refinement, orientation estimation, and category classification.

3.1. Generating Feature Maps From Point Clouds And Images

We follow the procedure described in [3] to generate a six-channel BEV map from a voxel grid representation of the point cloud at 0.1 meter resolution. The point cloud is cropped at $[-40, 40] \times [0, 70]$ meters to contain points

within the field of view of the camera. The first 5 channels of the BEV map are encoded with the maximum height of points in each grid cell, generated from 5 equal slices between $[0, 2.5]$ meters along the Z axis. The sixth BEV channel contains point density information computed per cell as $\min(1.0, \frac{\log(N+1)}{\log 16})$.

The proposed architecture uses two feature extractors, one for each input view. The feature extractors are based on the VGG16 architecture [25], with the following modifications. First, we use half the number of filters at every convolutional layer. Second, we use the method of [26] to initialize weights instead of using pre-trained ImageNet weights. Third, batch normalization is employed to provide similarly scaled feature maps from both views. Finally, we discard the fourth maxpooling layer and fifth convolutional layers of VGG16, resulting in output feature maps that have $8\times$ smaller resolution than their corresponding input, with a depth of 256. The output of the feature extraction from both views is passed through a $4\times$ bilinear upsampling layer to attain higher resolution feature maps. A visual representation of the feature extractors can be seen in Fig. 2.

3.2. Multimodal Fusion Region Proposal Network

Similar to 2D two-stage detectors, the proposed RPN regresses the difference between a set of prior 3D boxes and the ground truth. These prior boxes are referred to as anchors, and are encoded using the *axis aligned* bounding box encoding shown in Fig. 3. Anchor boxes are parameterized by the centroid (t_x, t_y, t_z) and axis aligned dimensions (d_x, d_y, d_z). To generate the 3D anchor grid, (t_x, t_y) pairs are sampled at an interval of 0.5 meters in BEV, while t_z is determined based on the sensor's height above the ground plane. Similar to [7], the dimensions of the anchors are determined by clustering the training samples for each class. Anchors without 3D points in BEV are removed efficiently via integral images resulting in $80 - 100$K non-empty anchors per frame.

Extracting Feature Crops Via Multiview Crop And Resize Operations: To extract feature crops for every anchor from the view specific feature maps, we use the crop and resize operation [27]. Given an anchor in 3D, two regions of interest are obtained by projecting the anchor onto the BEV and image feature maps. The corresponding regions are then used to extract feature map crops from each view, which are then bilinearly resized to 3×3 to obtain equal-length feature vectors.

Dimensionality Reduction Via 1×1 Convolutional Layers: In some scenarios, the region proposal network is required to save feature crops for 100K anchors in GPU memory. Attempting to extract feature crops directly from the 256 dimensional output feature maps imposes a large memory overhead per input view. As an example, extracting 7×7 feature crops for 100K anchors requires around 5 gigabytes[1] of memory assuming 32-bit floating point representation. Furthermore, processing such high-dimensional feature crops with the RPN greatly increases its computational requirements.

Inspired by their use in [28], we propose to apply a 1×1 convolutional kernel on the output feature maps from each view, as an efficient dimensionality reduction mechanism. The 1×1 convolution acts on every pixel position in each feature map according to:

$$f_{out} = \sigma(\sum_{i=0}^{256} w_i f_i + b), \tag{1}$$

where f_i is the pixel value at each one of the 256 feature map channels, w_i is a learned weight, and b is a learned bias term. These 1×1 convolutions can be thought of as strictly linear coordinate-dependent transformations in the filter space, followed by a non-linear activation function σ. By learning these transformations, the RPN reduces the dimensionality of its input feature maps while retaining information deemed useful for the proposal generation task. From a computational point of view, 1×1 convolutions are dot products across the depth of the feature map and can be computed efficiently on GPUs. The result is a feature selection mechanism that reduces the memory overhead for computing anchor specific feature crops by $256\times$ per view, allowing the RPN to process fused features of tens of thousands of anchors using only a few megabytes of additional memory.

3D Proposal Generation: The outputs of the crop and resize operation are equal-sized feature crops from both views, which are fused via an element-wise mean operation. Two task specific branches [16] of fully connected layers of size 256 use the fused feature crops to regress axis aligned object proposal boxes and output an object/background "objectness" score. 3D box regression is performed by computing $(\Delta t_x, \Delta t_y, \Delta t_z, \Delta d_x, \Delta d_y, \Delta d_z)$, the difference in centroid and dimensions between anchors and ground truth bounding boxes. Smooth L1 loss is used for 3D box regression, and cross-entropy loss for "objectness". Similar to [16], background anchors are ignored when computing the regression loss. Background anchors are determined by calculating the 2D IoU in BEV between the anchors and the ground truth bounding boxes. For the *car* class, anchors with IoU less than 0.3 are considered background anchors, while ones with IoU greater than 0.5 are considered object anchors. For the *pedestrian* and *cyclist* classes, the object anchor IoU threshold is reduced to 0.45. To remove redundant proposals, 2D non-maximum

[1] $100,000 \times 7 \times 7 \times 256 \times 4$ bytes.

suppression (NMS) at an IoU threshold of 0.8 is used to keep the top 1024 proposals during training. At inference time, 300 proposals are used for the car class, whereas 1024 proposals are kept for pedestrians and cyclists. These values are deduced from the recall vs number of proposals curves (Fig. 4) to obtain the highest possible recall for every class with the least number of proposals.

3.3. Second Stage Detection Network

3D Bounding Box Encoding: In [3], Chen et al. claim that 8 corner box encoding provides better results than the traditional axis aligned encoding previously proposed in [8]. However, an 8 corner encoding does not take into account the physical constraints of a 3D bounding box, as the top corners of the bounding box are forced to align with those at the bottom. To reduce redundancy and keep these physical constraints, we propose to encode the bounding box with four corners and two height values representing the top and bottom corner offsets from the ground plane, determined from the sensor height. Our regression targets are therefore $(\Delta x_1 ... \Delta x_4, \Delta y_1 ... \Delta y_4, \Delta h_1, \Delta h_2)$, the corner and height offsets from the ground plane between the proposals and the ground truth boxes. To determine corner offsets, we correspond the closest corner of the proposals to the closest corner of the ground truth box in BEV. The proposed encoding reduces the box representation from an overparameterized 24 dimensional vector to a 10 dimensional one.

Explicit Orientation Vector Regression: To determine orientation from a 3D bounding box, MV3D [3] relies on the extents of the estimated bounding box where the orientation vector is assumed to be in the direction of the longer side of the box. This approach suffers from two problems. First, this method fails for detected objects that do not always obey the rule proposed above, such as pedestrians. Secondly, the resulting orientation is only known up to an additive constant of $\pm\pi$ radians. Orientation information is lost as the corner order is not preserved in closest corner to corner matching. Fig. 1 presents an example of how the same rectangular bounding box can contain two instances of an object with opposite

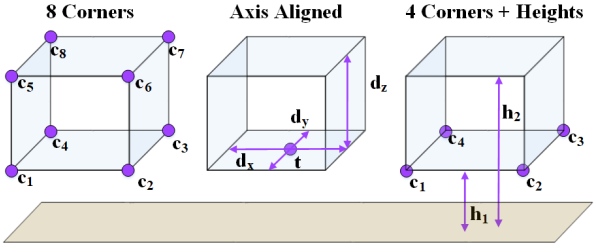

Figure 3: A visual comparison between the 8 corner box encoding proposed in [3], the axis aligned box encoding proposed in [8] and our 4 corner encoding.

orientation vectors. Our architecture remedies this problem by computing $(x_{or}, y_{or}) = (\cos(\theta), \sin(\theta))$. This orientation vector representation implicitly handles angle wrapping as every $\theta \in [-\pi, \pi]$ can be represented by a unique unit vector in the BEV space. We use the regressed orientation vector to *correct* the bounding box orientation estimate from the adopted four corner representation. Specifically, we extract the four possible orientations of the bounding box, and then choose the one closest to the explicitly regressed orientation vector. In Section 5, this process is shown to provide a more accurate orientation estimate than directly using the regressed orientation vector.

Generating Final Detections: Similar to the RPN, the inputs to the multiview detection network are feature crops generated from projecting the proposals into the two input views. As the number of proposals is an order of magnitude lower than the number of anchors, the original feature map with a depth of 256 is used for generating these feature crops. Crops from both input views are resized to 7×7 and then fused with an element-wise mean operation. A *single* set of three fully connected layers of size 2048 process the fused feature crops to output box regression, orientation estimation, and category classification for each proposal. Similar to the RPN, we employ a multi-task loss combining two Smooth L1 losses for the bounding box and orientation vector regression tasks, and a cross-entropy loss for the classification task. Proposals are only considered in the evaluation of the regression loss if they have at least a 0.65 or 0.55 2D IoU in BEV with the ground truth boxes for the *car* and *pedestrian/cyclist* classes, respectively. Since multiple proposals can be regressed to the same space in BEV, we perform 2D NMS with an IoU threshold of 0.01 to eliminate overlapping detections.

3.4. Training

We train two networks, one for the *car* class and one for both the *pedestrian* and *cyclist* classes. The RPN and the detection networks are trained jointly in an end-to-end fashion using mini-batches containing one image with 512 and 1024 ROIs, respectively. The network is trained for 120K iterations using an ADAM optimizer [29] with an initial learning rate of 0.0001 that is decayed exponentially every 30K iterations with a decay factor of 0.8. For regularization, we employ dropout with a probability of 0.5 throughout all the fully connected layers.

4. The Average Heading Similarity Metric

As discussed in Section 1, previously proposed evaluation metrics do not reliably capture the error in global orientation estimation for 3D object detectors. Inspired by the *Average Orientation Similarity* (AOS) [1] metric used by KITTI to jointly evaluate the 2D detection performance and

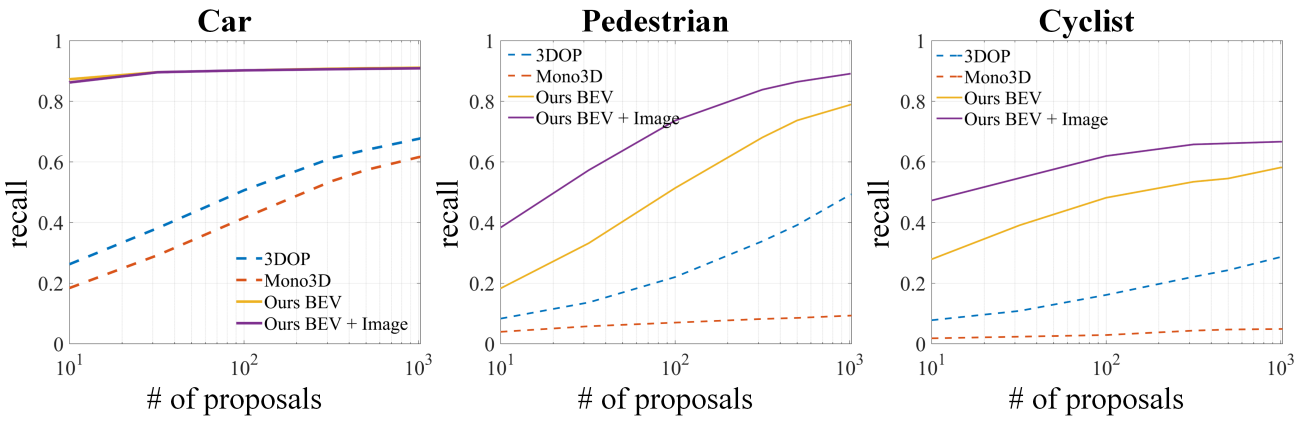

Figure 4: The recall vs number of proposals at a 3D IoU threshold of 0.5 for the three classes evaluated on the *validation* set at *moderate* difficulty.

observation angle estimate of a detector, we propose the *Average Heading Similarity* (AHS) to capture both the detectors localization performance *in 3D*, and its error in global orientation estimation. The AHS is calculated as:

$$AHS = \frac{1}{11} \sum_{r \in (0, 0.1, \ldots 1)} \max_{\hat{r}: \hat{r} \geq r} s(\hat{r}), \quad (2)$$

where $s(r)$ is defined at every recall value r according to:

$$s(r) = \frac{1}{|\mathbb{D}(r)|} \sum_{i \in \mathbb{D}(r)} \mathbb{1}(IoU \geq \lambda) \frac{1 + \cos(\theta - \theta^*)}{2}. \quad (3)$$

Here, θ is the global orientation estimate, θ^* is the ground truth global orientation, and $\mathbb{D}(r)$ is the set of all detections at recall r. $\mathbb{1}(IOU \geq \lambda)$ is an indicator function used to only consider valid detections in the computation of the heading similarity. The AHS can be evaluated in either 3D or BEV spaces by switching the source of the IoU computation in the indicator function. Code for evaluating the AHS is available here.

5. Experiments and Results

We test AVOD's performance on the proposal generation and object detection tasks on the three classes of the KITTI Object Detection Benchmark [1]. We follow [3] to split the provided 7481 training frames into a *training* and a *validation* set at approximately a $1:1$ ratio. For submission to the KITTI server, we train on our custom split with 85% of training data used for training and the rest for validation. For evaluation, we follow the *easy, medium, hard* difficulty classification proposed by KITTI.

3D Proposal Recall: 3D proposal generation is evaluated using *3D bounding box recall* at a 0.5 3D IoU threshold. We compare our RPN against the proposal generation algorithms 3DOP [17] and MONO3D [9]. Fig. 4 shows the recall vs number of proposals curves for our fusion RPN, 3DOP and Mono3D. It can be seen that our fusion based RPN outperforms both 3DOP and Mono3D by a wide margin on all three classes. As an example, our RPN achieves an 86% 3D recall on the *car* class with just 10 proposals per frame. The maximum recall achieved by 3DOP and Mono3D on the *car* class is 73.87% and 65.74% respectively. This gap is also present for the pedestrian and cyclist classes, where our RPN achieves more than 20% increase in recall at 1024 proposals. This large gap in performance suggests the superiority of learning based approaches over methods based on hand crafted features. For the car class, our RPN achieves a 91% recall at just 50 proposals, whereas MV3D [3] reported requiring 300 proposals to achieve the same recall. It has to be noted that MV3D does not provide proposal results for pedestrians or cyclists.

3D Object Detection 3D detection results are evaluated using the 3D AP, BEV AP, and AHS metrics at 0.7 IoU threshold for the car class, and 0.5 IoU threshold for the pedestrian and cyclist classes. We compare against publicly provided results from MV3D [3] and Deep3DBox [10] on the *validation* set. It has to be noted that no currently published method provides public results on the pedestrian and cyclist classes for the 3D object detection task, and

	Easy		Moderate		Hard	
	AP	AHS	AP	AHS	AP	AHS
Deep3DBox	5.84	5.84	4.09	4.09	3.83	3.83
MV3D	**83.87**	52.74	72.35	43.75	64.56	39.86
Ours	83.45	**83.42**	**74.19**	**73.97**	**67.78**	**67.44**

Table 1: A comparison of the performance of Deep3DBox [10], MV3D [3], and our method evaluated on the *car* class in the *validation* set. For evaluation, we show the AP and AHS (in $\%$) at 0.7 3D IoU.

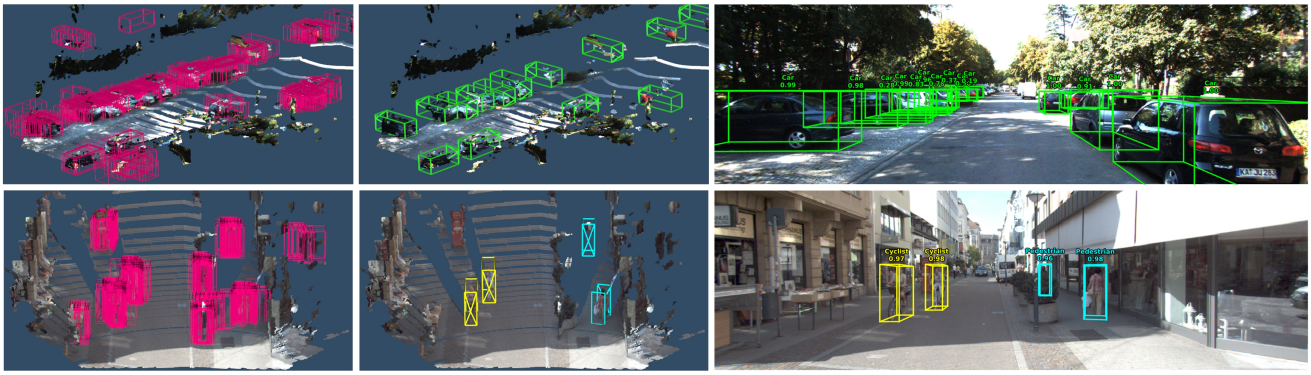

Figure 5: Qualitative results of AVOD for cars (top) and pedestrians/cyclists (bottom). **Left:** 3D region proposal network output, **Middle:** 3D detection output, and **Right:** the projection of the detection output into image. The 3D LIDAR point cloud has been colorized and interpolated for better visualization.

hence comparison is done for the *car* class only. On the *validation* set (Table 1), our architecture is shown to outperform MV3D by 1.84% AP on the *moderate* setting and 3.22% on the *hard* setting. However, AVOD achieves a 30.28% and 27.54% increase in AHS over MV3D at the *moderate* and *hard* setting respectively. This can be attributed to the loss of orientation vector direction discussed in Section 3.3 resulting in orientation estimation up to an additive error of $\pm\pi$ radians. To verify this assertion, Fig. 6 shows a qualitative comparison of the results of AVOD and MV3D in comparison to KITTI's ground truth. It can be seen that MV3D assigns erroneous orientations for almost half of the cars shown. On the other hand, our proposed architecture assigns the correct orientation for all cars in the scene. As expected, the gap in 3D localization performance between Deep3DBox and our proposed architecture is very large. It can be seen in Fig. 6 that Deep3DBox fails at accurately localizing most of the vehicles in 3D. This further enforces the superiority of fusion based methods over monocular based ones. We also compare the performance of our architecture on the KITTI *test* set with 3DFCN[5], MV3D, VoxelNet [21], and F-PointNet [23]. *Test* set results are provided directly by the KITTI evaluation server, which does not compute the AHS metric. Table 2 shows the results of AVOD on KITTI's *test* set. AVOD is shown to provide state-of-the-art results on all classes while being $2\times$ as fast as the top performing method, F-PointNet. MV3D, F-PointNet, and VoxelNet do not provide orientation estimation results to be evaluated using the 2D AOS metric. This further motivates the usage of the proposed AHS metric by KITTI's evaluation server for future evaluation.

Runtime and Memory Requirements: We use FLOP count and number of parameters to assess the computational efficiency and memory requirements of the proposed network. AVOD employs roughly 38.073 million parameters, approximately 37% that of MV3D. The *deep fusion* scheme employed by MV3D triples the number of fully connected layers required for the second stage detection network, which explains the two thirds reduction in the number of parameters by our proposed architecture. Furthermore, our proposed architecture requires 186.284 billion FLOPs per frame, allowing it to process frames in 0.08 seconds on a Titan Xp GPU. This makes it $4.5\times$ faster than MV3D, which requires more computation at the feature extraction stage due to the utilization of 3 input views. Finally, our proposed architecture requires only 2 gigabytes of GPU

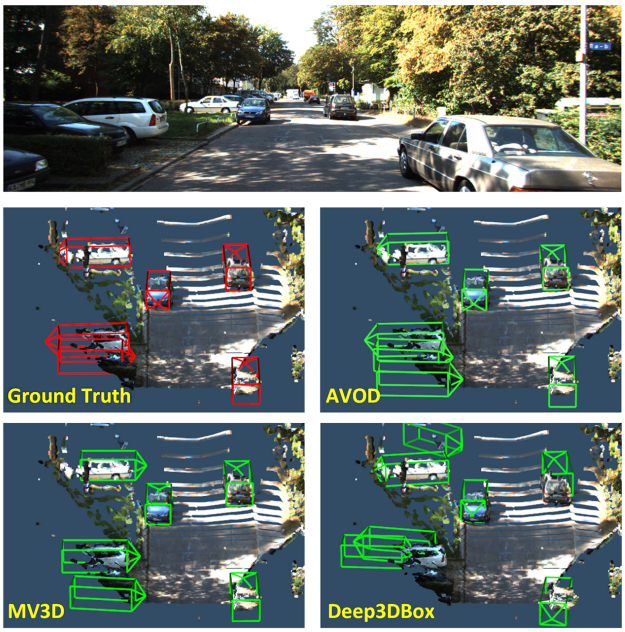

Figure 6: A qualitative comparison between MV3D [3], Deep3DBox [10], and our architecture relative to KITTI's ground truth on a sample in the *validation* set. It can be noted that MV3D can only determine the global orientation of objects with an additive error of $\pm\pi$ radians.

			AP_{3D}			AP_{BEV}			AP_{2D}			AOS_{2D}		
Method	Runtime (s)	Class	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
VeloFCN [4]	1	**Car**	-	-	-	0.15	0.33	0.47	-	-	-	-	-	-
3D FCN [5]	> 5		-	-	-	77.05	69.77	62.59	-	-	-	-	-	-
MV3D [3]	0.36		71.09	62.35	55.12	86.02	76.90	68.49	90.53	89.17	80.16	-	-	-
VoxelNet [21]	0.23		77.47	65.11	57.73	**89.35**	79.26	77.39	90.30	85.95	79.21	-	-	-
F-PointNet [23]	0.17		**81.20**	**70.39**	**62.19**	88.70	84.00	75.33	**90.78**	**90.00**	**80.80**	-	-	-
Ours	**0.08**		73.59	65.78	58.38	86.80	**85.44**	**77.73**	89.73	88.08	80.14	**89.59**	**87.46**	**79.54**
VoxelNet [21]	0.23	**Ped.**	39.48	33.69	31.51	46.13	40.74	38.11	50.61	44.08	42.84	-	-	-
F-PointNet [23]	0.17		**51.21**	**44.89**	**40.23**	**58.09**	**50.22**	**47.20**	**87.81**	**77.25**	**74.46**	-	-	-
Ours	**0.08**		38.28	31.51	26.98	42.51	35.24	33.97	51.64	43.49	37.79	**44.12**	**36.38**	**31.81**
VoxelNet [21]	0.23	**Cyc.**	61.22	48.36	44.37	66.70	54.76	50.55	72.04	59.33	54.72	-	-	-
F-PointNet [23]	0.17		**71.96**	**56.77**	**50.39**	**75.38**	**61.96**	**54.68**	**84.90**	**72.25**	**65.14**	-	-	-
Ours	**0.08**		60.11	44.90	38.80	63.66	47.74	46.55	65.72	56.01	48.89	**64.36**	**54.43**	**47.67**

Table 2: A comparison of the performance of AVOD with the state of the art 3D object detectors evaluated on KITTI's *test* set. Results are generated by KITTI's evaluation server [30].

memory at inference time, making it suitable to be used for deployment on autonomous vehicles.

5.1. Ablation Studies:

Table 3 shows the effect of varying different hyper-parameters on the performance measured by the AP and AHS, number of model parameters, and FLOP count of the proposed architecture. The original network uses hyperparameter values described throughout the paper up to this point. We study the effect of the RPN's input feature vector origin and size on both the proposal recall and final detection AP by training two networks, one using *BEV only* features, and the other using feature crops of size 1×1 as input to the RPN stage. We also study the effect of different bounding box encoding schemes shown in Fig. 3, and the effects of adding an orientation regression output layer on the final detection performance in terms of AP and AHS. Finally, we compare different fusion schemes proposed previously in [3]. The fusion schemes are shown in Fig. 7.

RPN Input Variations: Fig. 4 shows the recall vs number of proposals curves for both the original RPN and BEV only RPN on the three classes on the *validation* set. For the pedestrian and cyclist classes, fusing features from both views at the RPN stage is shown to provide a 10.1% and 8.6% increase in recall over the BEV only version at 1024 proposals. This difference increases as the number of proposals is decreased reaching 20% when using 100 proposals. For the car class, adding image features as an input to the RPN does not seem to provide a higher recall value over the BEV only version. We attribute this to the fact that instances from the car class usually occupy a large space in the input BEV map, providing sufficient features in the corresponding output feature map to reliably generate object proposals. The effect of the increase in proposal recall on the final detection performance can be observed in Table 3.

Using both image and BEV features at the RPN stage results in a 6.9% and 9.4% increase in AP over the BEV only version for the pedestrian and cyclist classes respectively. We perform an additional experiment using 1 pixel feature vector size as an input to the RPN from both views. We observe a drop in AP for all three classes, with the drop being much higher for the pedestrian and cyclist classes (26.9% and 32.6% decrease for pedestrians and cyclists vs a 9.5% decrease for cars). This enforces the hypothesis that a single pixel does not contain sufficient information for proposal generation for these smaller classes. It has to be noted that the BEV only RPN requires 266.64 million less FLOPs than the original network as it does not need to generate feature crops from image view feature maps. The decrease in the FLOP count for the network does not compensate for the loss in performance.

Bounding Box Encoding: We study the effect of different bounding box encodings shown in Fig. 3 by training two additional networks. The first network estimates axis

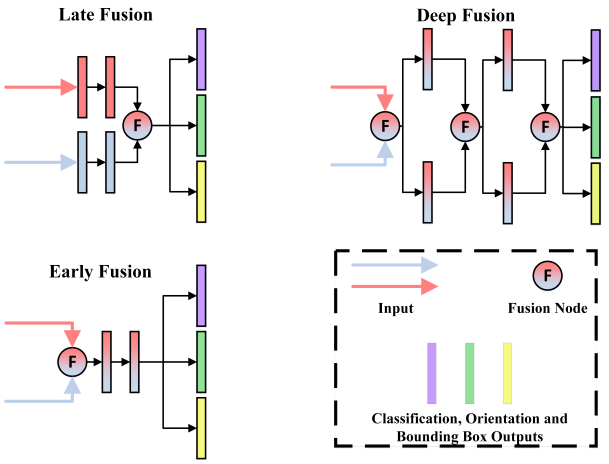

Figure 7: The different fusion schemes used for comparison in this paper. The element-wise *mean* operation is used for all fusion modes.

Architecture	Car		Pedestrian		Cyclist		Number Of Parameters	FLOPs
	AP_{3D}	AHS_{3D}	AP_{3D}	AHS_{3D}	AP_{3D}	AHS_{3D}		
Original	**74.1**	**73.9**	**39.5**	29.8	**41.6**	33.2	38,073,528	186,284,945,569
RPN BEV Only	**74.1**	**73.9**	32.6	26.7	32.2	30.2	0	**-266,641,569**
RPN 1 Pixel Feature Size	64.6	64.3	12.6	11.6	9.0	8.2	**-4,096**	-20,480
Axis Aligned	67.6	67.5	32.6	27.8	36.6	**35.6**	-8196	-40,958
4 Corners No Orientation	67.8	43.1	37.8	17.9	41.1	18.6	**-4,098**	-20,418
8 Corners No Orientation	66.9	34.1	37.8	18.1	40.4	21.0	+24638	+122,879
Late Fusion	73.2	72.6	38.0	**29.9**	36.8	34.8	+34,113,536	+170,536,962
Deep Fusion	72.6	72.4	36.4	27.6	36.9	32.2	+34,113,536	+170,536,962

Table 3: A comparison of the performance of different variations of hyperparameters, evaluated on the *validation* set at *moderate* difficulty. We use a 3D IoU threshold of 0.7 for the car class, and 0.5 for the pedestrian and cyclist classes. The effect of variation of hyperparameters on the FLOPs and number of parameters are measured relative to the original network.

aligned bounding boxes, using the regressed orientation vector as the final box orientation. The second and the third networks use our 4 corner and MV3D's 8 corner encodings without additional orientation estimation as described in Section 3.3. As expected, without orientation regression to provide orientation angle correction, the two networks employing the 4 corner and the 8 corner encodings provide a much lower AHS than the original network for all three classes. This phenomenon can be attributed to the loss of orientation information as described in Section 3.3.

Fusion Methods: Fig. 7 shows the different fusion methods implemented and compared in this work. The choice of using early fusion for our architecture was due to no observed improvement when using late or deep fusion. On the other hand, early fusion requires half the number of parameters in the second stage fully connected layers, and as such requires much less memory when deployed. Early fusion is also marginally faster at inference time.

Qualitative Results: Fig. 5 shows the output of the RPN and the final detections in both 3D and image space. To show generalization, we also deploy AVOD on our autonomous vehicle with a different set of cameras (Ximea MQ013CG-E2 1.3 MP) and a lower resolution LIDAR (Velodyne HDL-32e) than that used in KITTI. Results of generalization are shown in Fig. 8. More qualitative results including those of AVOD running in **snow** and **night** scenes are provided in video format here.

6. Conclusion

In this work we proposed AVOD, a 3D object detector for autonomous driving scenarios. The proposed architecture is differentiated from the state of the art by using a multimodal fusion RPN architecture, and is therefore able to produce accurate region proposals in road scenes, even for small classes. Furthermore, the proposed architecture employs explicit orientation vector regression to resolve the ambiguous orientation estimate inferred from a 3D bounding box. Experiments on the KITTI dataset show that our proposed architecture achieves state of the art results on the 3D localization, orientation estimation, and classification tasks. Finally, the proposed architecture is shown to run in real time and with a low memory overhead.

References

[1] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012. 1, 2, 5, 6

[2] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1

[3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, 2017. 1, 2, 3, 5, 6, 7, 8

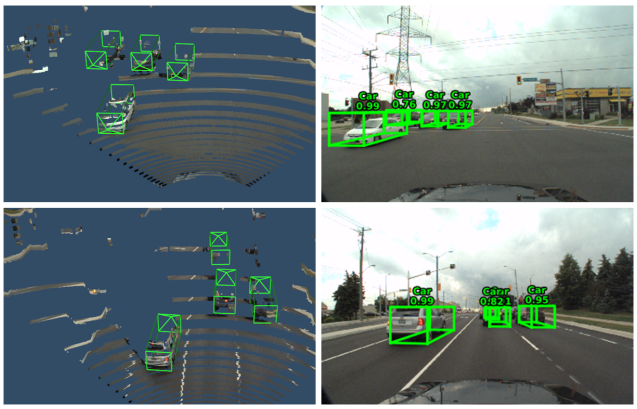

Figure 8: Results of deploying AVOD on our autonomous vehicle. It can be seen that the network generalizes well to new scenes even when using data from a different camera and a lower resolution LIDAR sensor.

[4] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016. 1, 2, 3, 8

[5] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *IROS*, 2017. 1, 2, 7, 8

[6] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1355–1361. IEEE, 2017. 1

[7] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals using stereo imagery for accurate object class detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 1, 4

[8] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016. 1, 3, 5

[9] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *IEEE CVPR*, 2016. 1, 2, 6

[10] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 6, 7

[11] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2

[12] Alejandro González, David Vázquez, Antonio M López, and Jaume Amores. On-board object detection: Multicue, multimodal, and multiview random forest of local experts. *IEEE transactions on cybernetics*, 2016. 1

[13] Michael Giering, Vivek Venugopalan, and Kishore Reddy. Multi-modal sensor registration for vehicle perception via deep neural networks. In *High Performance Extreme Computing Conference (HPEC), 2015 IEEE*, pages 1–6. IEEE, 2015. 1

[14] Cristiano Premebida, Joao Carreira, Jorge Batista, and Urbano Nunes. Pedestrian detection combining rgb and dense lidar data. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4112–4117. IEEE, 2014. 1

[15] Markus Enzweiler and Dariu M Gavrila. A multilevel mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 20(10):2967–2979, 2011. 1

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2, 3, 4

[17] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *NIPS*, 2015. 2, 6

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[19] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[20] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016. 2

[21] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *arXiv preprint arXiv:1711.06396*, 2017. 2, 7, 8

[22] Jean Lahoud and Bernard Ghanem. 2d-driven 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4622–4630, 2017. 3

[23] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017. 3, 7, 8

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 3

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[26] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 4

[27] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 4

[28] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 4

[29] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[30] Kitti 3D Object Detection Benchmark. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d. 8