IoU Loss for 2D/3D Object Detection

Dingfu Zhou[1,2], Jin Fang[1,2], Xibin Song[1,2], Chenye Guan[1,2],
Junbo Yin[3], Yuchao Dai[4] and Ruigang Yang[1,2]

[1]Baidu Research [2]National Engineering Laboratory of Deep Learning Technology and Application, China [3]Beijing Lab of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, China [4] Northwestern Polytechnical University, Xi'an, China

Abstract

In 2D/3D object detection task, Intersection-over-Union (IoU) has been widely employed as an evaluation metric to evaluate the performance of different detectors in the testing stage. However, during the training stage, the common distance loss (e.g., L_1 or L_2) is often adopted as the loss function to minimize the discrepency between the predicted and ground truth Bounding Box (Bbox). To eliminate the performance gap between training and testing, the IoU loss has been introduced for 2D object detection in [1] and [2]. Unfortunately, all these approaches only work for axis-aligned 2D Bboxes, which cannot be applied for more general object detection task with rotated Bboxes. To resolve this issue, we investigate the IoU computation for two rotated Bboxes first and then implement a unified framework, IoU loss layer for both 2D and 3D object detection tasks. By integrating the implemented IoU loss into several state-of-the-art 3D object detectors, consistent improvements have been achieved for both bird-eye-view 2D detection and point cloud 3D detection on the public KITTI [3] benchmark.

1. Introduction

Object detection, as a fundamental task in computer vision and robotics, has been well studied recently. For 2D object detection, many classical frameworks have been developed, including both two-stage methods (*e.g.*, fast R-CNN [4], faster R-CNN [5]) and one-stage methods (*e.g.*, SSD [6] and YOLO [7]). Recently, with the rapid development of the range sensors, such as the LiDAR and RGB-D cameras, 3D object detection has been attracting more and more researchers' attention. Similar with the 2D detection, some one- or two-stage based 3D object detection frameworks have been developed, such as Frustum-Pointnet [8], Voxel-net [9], SECOND [10], PointPillars [11] and Point R-CNN [12].

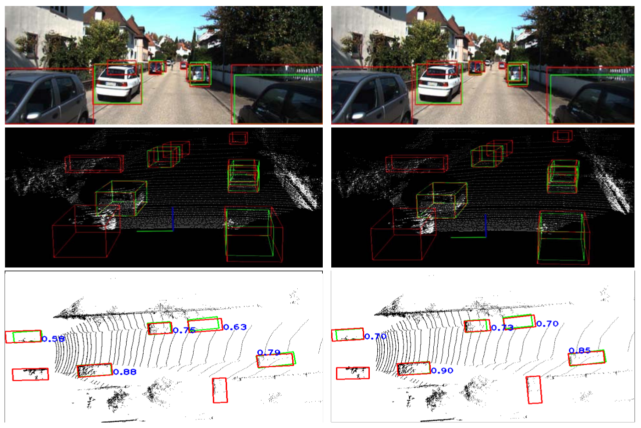

Figure 1. An example of 3D car detection results from different models trained with SECOND [10] and SECOND + proposed $\mathbf{L}_{IoU}$ loss are shown in the left and right columns. The IoU value for each Bbox has been provided in the Bird-eye-view image at the bottom this figure. From this figure, we can find that the accuracy of the bounding boxes has been steadily improved by using the proposed IoU loss.

For easy generalization, in the detection task objects are usually represented as 2D Bboxes or a 3D cuboids with several parameters, such as Bbox's center, dimension and orientation *etc.*. Therefore, object detection problem has been transformed as a regression task by minimizing the difference between ground truth Bbox and the predicted one. Currently, with superpower of the deep neural network, most of approaches focus on designing a better architecture backbone [9] or a better representation to extract the information of the foreground and background objects. For the loss function, they employed the common used L_1 and L_2 distance to optimize the whole network.

To compare the performance of different detectors, IoU metric is usually employed for evaluation, which is a total different metric compared with the L_1 and L_2 losses. As the name suggests, IoU (Intersection over Union) represents the area ratio of intersection to union of two shapes *e.g.*Bboxes. Compared with the L_1 and L_2 distance, the IoU metric has several advantages. First, all shape properties of the Bbox has been considered in the IoU computation process, *e.g.*, location, dimension and orientation *etc.*. Second, the area computation process has implicitly encoded the relationship between each parameter rather than considering them as independent variables in L_1 and L_2 loss. Finally, the IoU metric is scale invariant to the problem, which is suitable to solve the scale and range difference between each parameter.

Through the analysis above, we can clearly find that there is an obvious mismatch between the objective for model training and the metric for evaluation. Frankly speaking, there is no strong correlation between the L_1 loss and the IoU metrics. Two predicted Bboxes may have the same L_1 loss with the ground truth Bbox, while the IoU value of these two Bboxes could be totally different. To eliminate this kind of gap, some efforts have been made in [1] and [2] for 2D object detection.

Unfortunately, both of them are only suitable for the easy case with axis-aligned Bboxes and none of them can be applied for the general cases with two rotated Bboxes or 3D object detection. In this paper, we explored the IoU calculation between two rotated Bboxes first and then implemented a unified IoU loss function which can be used for both axis-aligned and rotated 2D object detection. In addition, the new IoU loss can be also applied for 3D object detection which has only one freedom of degree for orientation. The main contribution of this paper can be summarized as:

- We investigated the IoU loss computation for two rotated 2D and 3D Bboxes;
- We provided a unified, framework independent, IoU loss layer for general 2D and 3D object detection tasks.
- By integrating the IoU loss layer into several state-of-the-art 3D object detect frameworks such as SECOND, PointPillars and Point R-CNN, its superiority has been verified on the public KITTI 3D object detection benchmark.

2. Related Works

2.1. 2D object detection

Generic object detection frameworks can mainly be divided into two directions: the first direction is also called two-stage based methods, which generate region proposals at first stage and then classify each proposal into different classes. The other one is one-stage based methods which consider the object detection as a regress and classification problem by adopting a unified framework to obtain location and classes information simultaneously. R-CNN [13], Fast R-CNN [4], Faster R-CNN [5] and Mask R-CNN [14] are the most representative works of two-stages based methods, while MultiBox [15], YOLO [7], SSD [6], DSSD [16] are the representative works for one-stage based methods.

Although the design idea is slightly different between the one- and two-stage based framework, the Bbox parameters regression is a crucial component for both of them. For robust optimization and better regress results, different Bbox representation and loss functions have been designed. In YOLO [7], the authors proposed to directly regress the Bbox parameters for object detection. To solve the scale sensitivity, they proposed to predict square root of the bounding box size rather than itself. In R-CNN [13], the concept of prior Bbox which is also well known as proposals has been used. In this case, the Bbox regression can be transformed to predict the residual between the ground truth and the predicted Bboxes. Then L_2-norm is taken as the loss function for optimizing the framework. To against the outliers and noise, L_1-norm has been applied in Fast R-CNN [4]. After that the L_1-norm has been taken as a standard loss in the object detection frameworks [5, 14, 6].

2.2. 3D object detection

3D object detection in traffic scenario becomes more and more popular with the development of range sensor and the Autonomous Driving techniques. Inspired by 2D object detection, the point cloud is first projected into 2D (*e.g.*bird-eye-view [17] or front-view [18]) to obtained the 2D detection and then re-project the 2D Bbox into 3D to get the finally results. Another representative direction for 3D object detection is volumetric convolutional based methods due to the rapid development of the graphics processing resources. Voxel-net [9] is a pioneer work to detect the 3D objects directly with 3D convolutional by representing the LiDAR point cloud with voxels. For saving the GPU memory, the voxel resolution is relative large as $0.4\text{m} \times 0.2\text{m} \times 0.2\text{m}$. For each voxel, the PointNet [19] is applied to extract a 128-dimension features first. Based on the framework of Voxelnet, two variant methods, SECOND [10] and PointPillars [11] have been proposed. Different with the two directions mentioned above, PointNet [19] is another useful techniques for point cloud feature extraction. Along this direction, several state-of-the-art methods have been proposed for 3D object detection [8, 12]. Similar to the 2D object detection framework, the common L_1-norm has been employed directly for 3D Bbox regression.

2.3. IoU Loss for Object Detection

Most of the frameworks used a surrogate loss (*e.g.*, L_1 or L_2 distance loss) of IoU for Bbox regression. The drawbacks of this kind of loss function have been found in [1, 20] and [2]. In [1], a novel IoU loss function for axis-aligned bounding box prediction has been introduced, which regresses the four bounds of a predicted box as a whole unit, performs accurate and efficient localization, shows robust to objects of varied shapes and scales, and converges fast. In [20], bounded IoU loss has been developed, which is proved to be better matching the goal of IoU maximization while still providing good convergence properties. Furthermore, in [2], the authors discussed the weakness of IoU for the case of non-overlapping bounding boxes first and then introduced a generalized version of IoU (GIoU) as a new loss. Finally, the effectiveness of GIoU has been verified by integrating it into the state-of-the-art 2D object detection frameworks [5, 7, 14]. All the works mentioned above target on the axis-aligned Bbox regression task, none of the works have proposed to apply the IoU loss for rotated Bbox or 3D object detection tasks.

3. IoU for Object Detection

IoU is also known as the Jaccard index (or the Jaccard similarity coefficient) which has been widely used to measure the similarity between finite sample sets. Generally, for two finite sample sets $\mathbf{A}$ and $\mathbf{B}$, their IoU is defined as the intersection ($\mathbf{A} \cap \mathbf{B}$) divided by the union ($\mathbf{A} \cup \mathbf{B}$) of $\mathbf{A}$ and $\mathbf{B}$.

$$\mathbf{IoU}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cap \mathbf{B}}{\mathbf{A} \cup \mathbf{B}} = \frac{\mathbf{A} \cap \mathbf{B}}{|\mathbf{A}| + |\mathbf{B}| - \mathbf{A} \cap \mathbf{B}} \qquad (1)$$

As its definition in [2], IoU fulfills all properties of a metric, such as non-negativity, identity of indiscernibles, symmetry and triangle inequality. Especially, IoU is invariant to the scale which means that the similarity between two arbitrary shapes $\mathbf{A}$ and $\mathbf{B}$ is independent from the scale of their space. Due to these properties mentioned above, the IoU has been widely employed as evaluation metric for many task in computer vision, *e.g.*, pixel- or instance-level image segmentation, 2D/3D object detection *etc*.. Particularly, we only focus on the task of object detection and its application for other tasks is beyond the scope of this paper.

3.1. IoU Definition for Object Detection

For bounding box-level object detection, the target object is usually represented by a minimum Bbox rectangle in the 2D image. Base on this representation, the IoU computation between the ground bounding box $\mathbf{B}_g$ and the predicted bounding box $\mathbf{B}_d$ is defined as

$$\mathbf{IoU}(\mathbf{B}_g, \mathbf{B}_d) = \frac{\text{Aera of overlap } \mathbf{B}_g \text{ and } \mathbf{B}_d}{\text{Aera of union } \mathbf{B}_g \text{ and } \mathbf{B}_d}. \qquad (2)$$

For 3D object detection, the Bbox is simply replaced by a cuboid and the IoU value between two cuboids can be easily obtained by changing the area with volume in Eq. (2). For simplicity, we only take 2D case into the consideration here and its extension to 3D will be introduced in the following sections.

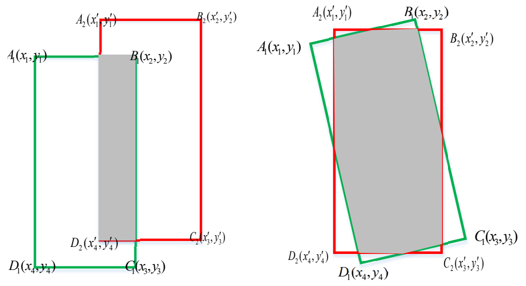

Figure 2. IoU computation for 2D: axis-aligned and rotated bounding boxes, where the green and red represent the ground truth and predicted bounding box respectively. The intersection area is highlighted in gray.

Algorithm 1 **IoU** for two axis-aligned BBoxes.

```
Require: -Corners of the two bounding boxes:
    $A_1(x_1, y_1), B_1(x_2, y_1), C_1(x_2, y_2), D_1(x_1, y_2)$,
    $A_2(x_1', y_1'), B_2(x_2', y_1'), C_2(x_2', y_2'), D_2(x_1', y_2')$,
    where $x_1 \le x_2, y_2 \le y_1$ and $x_1' \le x_2', y_2' \le y_1'$
Ensure: - IoU value;
1: ▶ The area of $\mathbf{B}_g$: $\mathbf{Area}_g = (x_2 - x_1) \times (y_1 - y_2)$;
2: ▶ The area of $\mathbf{B}_d$: $\mathbf{Area}_d = (x_2' - x_1') \times (y_1' - y_2')$;
3: ▶ The area of overlap: $\mathbf{Area}_{overlap} = (\max(x_2, x_2') - \min(x_1, x_1')) \times (\max(y_1, y_1') - \min(y_2, y_2'))$;
4: ▶ $\mathbf{IoU} = \frac{\mathbf{Area}_{overlap}}{\mathbf{Area}_g + \mathbf{Area}_d - \mathbf{Area}_{overlap}}$;
```

3.2. Axis-aligned BBox

Usually, objects are labeled with axis-aligned BBoxes in most of the 2D object detection benchmarks, such as Pascal Visual Object Classes (VOC) Challenge [21], COCO [22] and KITTI [3]. By taking this kind of labels as ground truth, the predicted Bboxes are also axis-aligned rectangles. For this case, the IoU computation is very easy, which can be implemented with some basic math functions, such as "max" and "min" *etc*.. The left of Fig. 2 illustrates an example of intersection between two axis-aligned Bboxes where the shadow area represents the intersection area. The pseudo-code of IoU computation for the axis-aligned case is given in Alg. 1.

3.3. Rotated BBox

However, the axis-aligned box is not suitable for representing the target objects in 3D, such as the objects in the LiDAR point cloud. Usually, the 3D object is represented by a 3D cuboid. For autonomous driving scenario, the general 3D BBox with three degree-of-freedoms for rotation can be reduced to one (*e.g.*, "yaw" angle) by assuming that all the objects should lay on a relative flat road ground. This kind of representation is widely used in most of the popular 3D object detection benchmarks, such as KITTI [3] and-nuScenes [23]. An example of labeled 3D object in KITTI data is given in Fig. 1.

For evaluation of different methods, two different strategies have been provided in KITTI: 2D Bbox overlap by projecting the 3D objects into the Bird-Eye-View (BEV) or 3D Bbox overlap directly. Here, we discussed the 2D case first and the 3D case is similar to the 2D case by adding a height dimension simply. In the BEV image, objects are represented with rotated BBoxes as described in the bottom of Fig. 1. The IoU computation for two rotated rectangles is more complex than axis-aligned ones because they can be intersected in many different ways. A typical example of intersection of two rotated rectangles is shown at the right of Fig. 2 and the overlap part is highlighted in blue. How to get the area of overlap part is the critical step for IoU computation. The pseudo-code for IoU computation with two rotated BBoxes is given in Alg. 2.

Algorithm 2 IoU for two rotated BBoxes.

```
Require: -Corners of the two bounding boxes:
Ensure: - IoU value;
1: ▶ Compute the area of $\mathbf{B}_g$: $\mathbf{Area}_g = \mathbf{a} \times \mathbf{b}$, where $\mathbf{a} = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$ and $\mathbf{b} = \sqrt{(x_2-x_3)^2+(y_2-y_3)^2}$;
2: ▶ Compute the area of $\mathbf{B}_d$: $\mathbf{Area}_d = \mathbf{a}' \times \mathbf{b}'$ where $\mathbf{a}' = \sqrt{(x_2'-x_1')^2+(y_2'-y_1')^2}$ and $\mathbf{b}' = \sqrt{(x_2'-x_3')^2+(y_2'-y_3')^2}$;
3: ▶ Determine the vertexes of overlap area if they have.
4: ▶ Sort these polygon vertexes in anticlockwise order;
5: ▶ Compute the intersection area $\mathbf{Area}_{overlap}$;
6: ▶ $\mathbf{IoU} = \frac{\mathbf{Area}_{overlap}}{\mathbf{Area}_g+\mathbf{Area}_d-\mathbf{Area}_{overlap}}$;
```

3.4. 3D Bboxes

As we have mentioned before, 3D object in autonomous driving is usually represented by a 3D Bbox with seven parameters, which are three for location, three for dimension and one for rotation. In this case, the IoU for two 3D Bboxes can be calculated as

$$\mathbf{IoU}_{3D} = \frac{\mathbf{Area}_{\text{overlap}} \times h_{\text{overlap}}}{(\mathbf{Area}_g \times h_{\text{g}} + \mathbf{Area}_d \times h_{\text{d}} - \mathbf{Area}_{\text{overlap}} \times h_{\text{overlap}})}, \quad (3)$$

where h_{overlap} and h_{union} represents the intersection and union in the height direction.

4. IoU Loss for 2D/3D BBox Regression

So far, we have introduced IoU as a metric for two 2D and 3D BBoxes evaluation. Recently, some pioneers have succeeded in integrating the IoU loss [1, 2] for BBox regression in popular 2D object detection frameworks [4, 14, 7]. Unfortunately, both of them can only handle two axis-aligned BBoxes and none of works have been proposed to deal with more general cases, such as two rotated BBoxes or 3D object detection. As we have discussed in the previous section, the computation of intersection between two rotated BBoxes is not trivial and there is not an off-the-shelf implementation in the existing deep-learning frameworks. To well rectify this situation, we first investigated the IoU loss for two rotated BBoxes and then implemented it as an unified loss layer for both 2D and 3D object detection frameworks.

4.1. IoU as Loss

In [1] and [2], the effectiveness of IoU as loss function has been well proved for 2D axis aligned BBox regression task. Theoretically, it should also work well for rotated BBox because the only difference is the computation process for rotated ones is more complex than axis-aligned ones. Similar with [2], we defined the IoU loss as

$$\mathbf{L}_{\text{IoU}} = 1 - \mathbf{IoU}. \quad (4)$$

Because $\mathbf{IoU}$ satisfies $0 \leqslant \mathbf{IoU} \leqslant 1$, then the $\mathbf{L}_{\text{IoU}}$ is also bounded between 0 and 1.

4.2. IoU Loss Layer

Currently, the IoU loss for two rotated Bboxes has not implemented in any deep learning frameworks. Therefore, we implement both the forward and backward operations for this IoU loss layer.

4.2.1 Forward

As described in Alg. 2, the forward process includes the following steps:

1. Compute the areas for $\mathbf{B}_d$ and $\mathbf{B}_g$, where $\mathbf{B}_d$ and $\mathbf{B}_g$ represent the predicted and ground truth BBoxes respectively;

2. Determine the vertexes of intersection area between $\mathbf{B}_d$ and $\mathbf{B}_g$, which come from two ways: one is from the intersections of two BBoxes' edges and the other is from the BBoxes' corner who is inside the other BBox. The IoU value is zero if the vertexes don't exist.

3. Theoretically, these vertexes form a convex hull. For computation the area of this convex hull, we need sort the vertexes in anticlockwise (or clockwise) order. First of all, the center point of these vertexes is computed. Then, the

rotation angle formed by each vertex and the center is calculated. Finally, the vertexes can be sorted by the rotation angles.

4. Then, the intersection area is obtained by dividing it into small individual triangles.

5. Compute the IoU value based on Eq. (2) and the $\mathbf{L}_{\text{IoU}}$ via Eq. (4).

4.2.2 Backward

Currently, the derivative of common functions has been implemented in most of the public deep learning frameworks and the back-propagation process can be automatically triggered by calling these derivative computation functions. However, the analytical solution of the IoU calculation process is not easy to be provided due to the complexity of intersection between two rotated Bboxes. Especially, there exist some custom operations (intersection of two edges and sorting the vertexes *etc*.) whose derivative functions have not been implemented in the existing deep learning frameworks. Finally, we implement the backward operations for all these functions and we will make the source code public in the future.

4.2.3 Extension to GIoU Loss

As a generalized version of IoU, GIoU has been proposed in [2] to handle the case that two shapes don't have an intersection. In GIoU, a definition has been given to determine the distance between two non-intersected Bboxes. Generally speaking, for any two convex shapes $\mathbf{A}$, $\mathbf{B}$, a minimum area bounding shape $\mathbf{C}$ is defined as: the smallest convex shapes enclosing both $\mathbf{A}$ and $\mathbf{B}$. Usually, $\mathbf{C}$ should shares the same shape type with $\mathbf{A}$ and $\mathbf{B}$ for easy computation. Finally, the GIoU is defined as

$$\mathbf{GIoU} = \mathbf{IoU} - \frac{\mathbf{Area_C} - \mathbf{U}}{\mathbf{Area_C}}, \tag{5}$$

where $\mathbf{U} = \mathbf{Area_A} + \mathbf{Area_B} - \mathbf{Area}_{overlap}$. Similar as IoU loss, we also extended the GIoU loss for the case of rotated Bboxes.

5. Experimental Results

The proposed loss layer is an framework independent modular which can be integrated into any regression-based 2D or 3D object detection methods. Different with [2], the proposed loss layer is more general on both axis-aligned and non-axis-aligned cases, such as 2D BEV or 3D object detection. We integrate the proposed IoU/GIoU loss on different types of 3D object detection frameworks and then compare their performances on the public third-party 3D object detection benchmark.

Baselines: three state-of-the-art 3D object detectors have been evaluated here: SECOND[10], PointPillars [11] and PointRCNN [12]. SECOND is a voxel-based one-stage object detector, which is an advance version of VoxelNet [9] by adding a sparse convolution operations implemented by themself. PointPillars is an acceleration version of SECOND which represents the point cloud by pillars rather than voxels. First, PointNet[19] is employed to extract features for each "Pillar" and then the "Pillar" is taken as the minimum elements for the further convolution network. Compared with SECOND, the pillar expression is much faster than voxel representation. Different with the previous two methods, PointRCNN is a two-stage 3D object detector, which combines the point segmentation and region proposal at the first stage and the Bbox refinement is executed at the second stage of the framework.

Dataset: we train all the baselines and evaluate them on KITTI [3] 3D object detection benchmark. This data has been divided into training and testing two subsets, which consists of 7481 and 7518 frames respectively. Since the ground truth for the testing set is not available, we subdivide the training data into a training and validation set as described in [9, 10]. Finally, we obtained 3,712 data samples for training and 3,769 data samples for validation. On the KITTI benchmark, the objects have been categorized into "easy", "moderate" and "hard" based on their height in the image and occlusion ratio, etc. For each frame, both the camera image and the LiDAR point cloud has been provided, while only the point cloud has been used for our object detection here and the RGB image is only used for visualization.

Evaluation protocol: In this paper, we employ similar evaluation metric as KITTI [3] to report all our results. In [3], all the objects have been divided into "Easy", "Moderate" and "Hard" category based on their distances and occlusion ratios. For each category, we calculate the Average precision (AP) by giving a certain IoU threshold. Different with KITTI, we set three different thresholds here. Beside this, we also give the mean Average Precision (mAP) across different value of IoU thresholds, i.e. $\mathbf{IoUs} = \{0.50, 0.55, \ldots, 0.90, 0.95\}$ to evaluate the performance of detectors at different thresholds.

5.1. SECOND [10]

Training protocol: the officially released code [1] by the authors has been used for training the baseline SECOND model. We use exactly the same config file provided by the author and follow the same training protocol to achieve the baseline results on the KITTI benchmark. Compared with the baseline network, we just simply replace the regress loss $\mathbf{L}_1$ with our self-implemented $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ losses. We used nearly the same training strategy as the baseline *e.g.*,

[1] https://github.com/traveller59/second.pytorch

Loss Types	AP70			AP75			AP80			mAP		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
SECOND[10] + $\mathbf{L}_1$	88.15	78.33	77.25	81.37	66.86	65.48	59.56	48.90	44.45	62.41	57.52	56.23
SECOND + $\mathbf{L}_{\text{IoU}}$	**89.16**	78.99	77.78	**83.40**	**73.36**	**66.72**	**66.36**	**52.60**	**50.61**	**64.55**	**58.96**	**57.61**
Rel improvement ⇑	0.94%	0.82%	0.91%	**2.49%**	**9.72%**	**1.89%**	**11.42%**	**7.57%**	**13.86%**	**3.43%**	**2.50%**	**2.45%**
SECOND + $\mathbf{L}_{\text{GIoU}}$	89.15	**79.14**	**78.11**	82.56	72.98	66.34	64.27	51.67	50.11	64.38	58.73	57.20
Rel improvement ⇑	**1.13%**	**1.03%**	**1.11%**	1.46%	9.15%	1.31%	7.91%	5.66%	12.73%	3.16%	2.10%	1.72%

Table 1. Evaluation results by training SECOND [10] with $\mathbf{L}_1$ loss and proposed losses on validation dataset of the KITTI 3D car detection benchmark. All the numbers are the higher the better. The best result of each column has been highlighted with bold font.

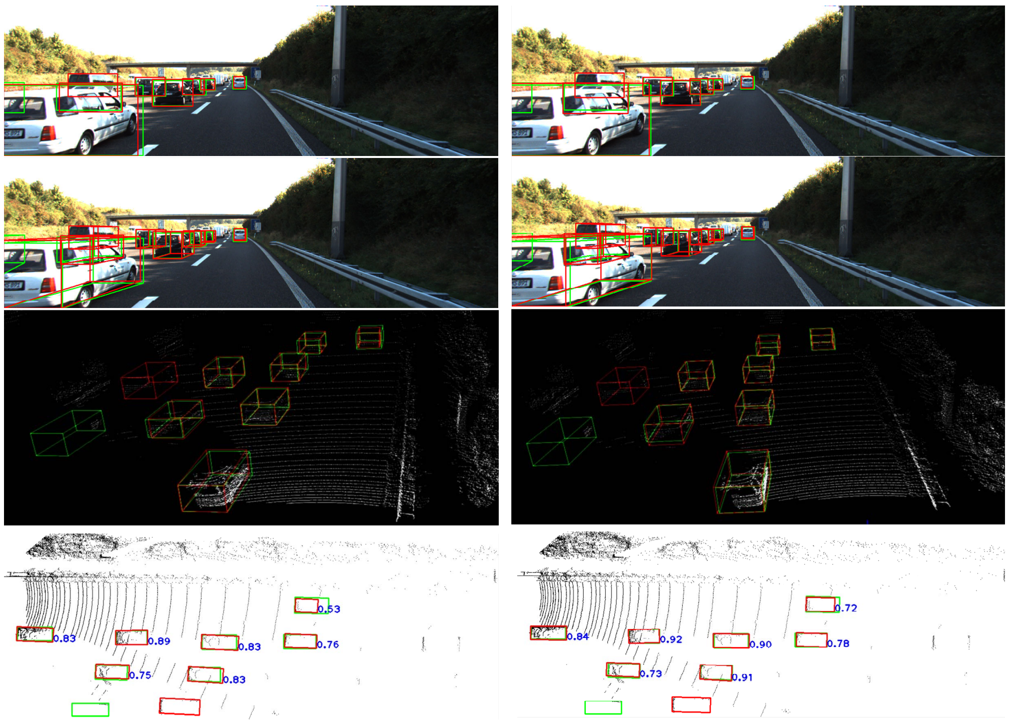

Figure 3. An example of 3D car detection results with different methods, where the left is from original SECOND method and the right is the SECOND with the proposed IoU loss.

Loss Types	BEV(AP70)			mAP		
	Easy	Mod	Hard	Easy	Mod	Hard
SECOND[10] + $\mathbf{L}_1$	89.92	87.88	86.72	70.63	67.71	65.82
SECOND + $\mathbf{L}_{\text{IoU}}$	90.21	88.25	87.56	71.39	68.37	66.23
Rel improvement ⇑	0.32%	0.42%	0.97%	1.07%	0.97%	0.62%
SECOND + $\mathbf{L}_{\text{GIoU}}$	**90.25**	**88.51**	**87.65**	**73.35**	**68.48**	**66.92**
Rel improvement ⇑	**0.37%**	**0.72%**	**1.07%**	**3.85%**	**1.14%**	**1.67%**

Table 2. Evaluation results of SECOND [10] with $\mathbf{L}_1$ and IoU losses on validation dataset of the KITTI BEV car detection benchmark. The number is the higher the better. The best result of each column is highlighted with bold font.

same iteration steps and learning rate *etc*.. The only difference is that we decrease the threshold of an anchor be considered as a positive sample during the training from 0.6 to 0.5, which means that there are more positive anchors have been involved in the training process. We set threshold at 0.6 in baseline framework because it gives better results than 0.5.

Results: The comparison of the IoU losses with original SECOND method for 3D car detection on KITTI benchmark has been given in Tab. 1. On this benchmark, the matching IoUs threshold 0.7 is used to evaluation, however, as mentioned by [20], the hyper-parameters (e.g., the matching IoUs threshold) usually have big influences on the detectors if only a certain matching IoUs threshold. Therefore, three different matching thresholds $\{0.70, 0.75, 0.80\}$ and the mAP have been applied here for evaluation.

From this table, we can find that the proposed $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ gives slightly better results than baseline for all the three categories ("easy", "moderate" and "hard") at the IoUs matching threshold 0.7. Compared with the baseline, around 1% relative improvement has been given by $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ for the three categories. At this threshold, the $\mathbf{L}_{\text{GIoU}}$ performs slightly better than $\mathbf{L}_{\text{IoU}}$.

We also find an interesting phenomenon that the $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ losses give much more improvements than base-

Loss Types	AP70			AP75			AP80			mAP		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
PointPillars [11] + $\mathbf{L}_1$	87.29	76.99	70.84	72.39	62.73	56.40	47.23	40.89	36.31	58.62	54.86	52.74
PointPillars+ $\mathbf{L}_{\text{IoU}}$	87.88	77.92	75.70	76.18	65.83	62.12	**57.82**	**45.03**	**42.95**	**62.07**	**57.11**	**55.67**
Rel improvement⇑	0.68%	1.21%	6.86%	5.24%	4.94%	10.14%	**22.4%**	**10.1%**	**18.28%**	**5.89%**	**4.10%**	**5.56%**
PointPillars + $\mathbf{L}_{\text{GIoU}}$	**88.43**	**78.15**	**76.34**	**76.93**	**66.36**	**63.68**	56.36	44.43	42.72	61.94	56.65	55.13
Rel improvement⇑	**1.34%**	**1.47%**	**7.62%**	**6.27%**	**5.78%**	**12.9%**	19.3%	8.66%	17.65%	5.53%	2.44%	4.17%

Table 3. Evaluation results by training PointPillar [11] with $\mathbf{L}_1$ loss and proposed losses on validation dataset of the KITTI 3D car detection benchmark. All the numbers are the higher the better. The best result of each column has been highlighted with bold font.

line when the IoUs matching threshold at a higher value. We can see clearly that the improvements at **AP80** are much greater than **AP70**. At **AP80**, the relative improvements for the three categories can reach **11.42%**, **7.57%** and **13.86%** respectively by using the $\mathbf{L}_{\text{IoU}}$ loss. At this threshold, the improvements for $\mathbf{L}_{\text{GIoU}}$ loss can achieve to 7.91%, 5.66% and 12.73% which performs slightly worse than $\mathbf{L}_{\text{IoU}}$.

The mAPs for all methods have been given in the last column of this table. We can also easily find that the detection performance has been steadily improved by the $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ losses. By using the new loss, all the detection rates have an average improvement of 2% and the improvement can reach 3% for some specific category.

Loss Types	BEV(AP70)			mAP		
	Easy	Mod	Hard	Easy	Mod	Hard
PointPillars [11] + $\mathbf{L}_1$	90.07	87.06	83.81	69.11	66.84	65.36
PointPillars + $\mathbf{L}_{\text{IoU}}$	90.24	88.02	86.64	71.33	**68.11**	66.53
Rel improvement ⇑	0.19%	1.10%	3.38%	3.21%	**1.90%**	1.79%
PointPillars + $\mathbf{L}_{\text{GIoU}}$	**90.35**	**88.26**	**87.04**	**71.74**	68.04	**66.63**
Rel improvement ⇑	**0.31%**	**1.37%**	**3.85%**	**3.81%**	1.80%	**1.94%**

Table 4. Evaluation results by training PointPillars [11] with $\mathbf{L}_1$ loss and proposed losses on KITTI validation dataset for BEV image. The best result of each column has been highlighted with bold font.

The detection results of BEV image is given in Tab. 2. Compared with the baseline, we can also find that the detection rate has been slightly improved with the proposed IoU loss for all the three categories. An example of detection results in BEV image and point cloud is given in Fig. 3. The bottom of this figure gives the 2D detection in BEV image, where the number around each Bbox is the IoU value in 3D. We can found that most of the values in right is larger than left, which means that the bounding box's accuracy has been consistently improved by the proposed IoU loss.

5.2. PointPillar [11]

Training protocol: the officially released code [2] by the authors has been used for training the PointPillars baseline model. We reproduce the baseline results on the KITTI benchmark, following the officially configure file and training protocols. Similar with SECOND method, we replaced the regression loss with our proposed $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ losses and decrease the foreground threshold from 0.6 to 0.5.

Results: The comparison of the proposed losses with original PointPillars is given in Tab. 3. The similar evaluation criterion is applied here too. From Tab. 3, the power of the proposed losses is demonstrated clearly. For **AP70**, the detection rates have been improved around 1% for "easy" and "moderate" categories, 6.86% and 7.63% for "hard" category with $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ losses respectively. For **AP80**, the proposed $\mathbf{L}_{\text{IoU}}$ loss can achieve a significant improvement by **22.4%**, **10.1%**, **18.28%** on the three categories, for the proposed $\mathbf{L}_{\text{GIoU}}$, which performs slightly worse, but also inspiring, promoted the baseline by 19.3%, 8.66%, 17.65% respectively.

The mAP values are shown in the last column, which have been steadily improved by over 4% roughly for both the $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ losses compared with the baseline. And the improvement can reach 5% for some specific categories.The detection results on the BEV images are shown in Tab. 4. From this table, we can also find steadily improvement with the proposed $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ losses.

5.3. PointRCNN [12]

Training protocol: different with the previous two methods, PointRCNN is a two-stage based framework. At the first stage, RPN network is employed to generate region proposal first and the Bbox refinement is executed at the second stage to get the final results. We trained the baseline with the official released source code here [3]. Currently, we kept the RPN part unchanged and integrated the proposed IoU loss only at the second stage. The loss function in the stage includes class classification loss, bin classification loss and the regression loss. Here, we keep the first two parts unchanged and replace the regression loss with the proposed $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$. To be clear, based on the officially released code, we cannot obtain the results reported in their paper. Therefore, the baseline reported here is the best model that we can achieve. Particularly, we train the baseline and the proposed losses by using the same training strategies for fair comparison.

Results: the comparison of PointRCNN with different losses is given in Tab. 5. Similar to the previous methods, we can easily find that both the $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ can improve baseline's performance at different IoU threshold for all the

[2] https://github.com/nutonomy/second.pytorch

[3] https://github.com/sshaoshuai/PointRCNN

Loss Types	**AP70**			**AP75**			**AP80**			**mAP**		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
PointRCNN [12]	88.14	77.58	75.36	73.27	63.54	61.08	44.21	38.88	34.62	59.44	54.35	52.79
PointRCNN+ $\mathbf{L}_{\text{IoU}}$	88.83	78.80	**78.18**	77.42	67.83	66.85	58.22	49.09	45.38	**63.47**	57.71	56.67
Rel improvement⇑	0.78%	1.57%	**3.74%**	5.66%	6.75%	9.44%	31.6%	26.26%	31.08%	**6.78%**	6.18%	7.35%
PointRCNN + $\mathbf{L}_{\text{GIoU}}$	**88.84**	**78.85**	78.15	**77.47**	**67.98**	**67.18**	**59.80**	**51.25**	**46.50**	63.12	**57.96**	**56.92**
Rel improvement⇑	**0.79%**	**1.64%**	3.70%	**5.73%**	**6.99%**	**9.99%**	**35.3%**	**31.81%**	**34.31%**	6.19%	**6.64%**	**7.82%**

Table 5. Evaluation results by training PointRCNN [12] with $\mathbf{L}_1$ loss and proposed losses on KITTI validation dataset for BEV image. The best result of each column has been highlighted with bold font.

Methods	Modality	**AP70**		
		Easy	Mod	Hard
MV3D[24]	LiDAR+Mono	71.29	62.68	56.56
F-PointNet[8]	LiDAR+Mono	83.76	70.92	63.65
AVOD-FPN[25]	LiDAR+Mono	84.41	74.44	68.65
ContFusion[26]	LiDAR+Mono	86.33	73.25	67.81
IPOD [27]	LiDAR+Mono	84.10	76.40	75.30
F-ConvNet[28]	LiDAR+Mono	89.02	78.80	77.09
VoxelNet[9]	LiDAR	81.97	65.46	62.85
PointPillars [11]	LiDAR	87.29	76.99	70.84
PointRCNN[12]	LiDAR	88.88	78.63	77.38
SECOND[10]	LiDAR	88.15	78.33	77.25
SECOND+$\mathcal{L}_{\text{IoU}}$	LiDAR	**89.16**	***78.99***	***77.78***
SECOND+$\mathcal{L}_{\text{GIoU}}$	LiDAR	***89.15***	**79.14**	**78.11**

Table 6. Comparison with other public methods on the KITTI validation dataset for 3D "Car" detection. For easy understanding, we have highlighted the top two numbers in bold and italic for each column. All the numbers are the higher the better.

Methods	Modality	**AP70**		
		Easy	Mod	Hard
MV3D[24]	LiDAR+Mono	71.09	62.35	55.12
F-PointNet[8]	LiDAR+Mono	81.20	70.29	62.19
AVOD-FPN[25]	LiDAR+Mono	81.94	71.88	66.38
ContFusion[26]	LiDAR+Mono	82.54	66.22	64.04
IPOD [27]	LiDAR+Mono	79.75	72.57	66.33
F-ConvNet[28]	LiDAR+Mono	***85.88***	**76.51**	68.08
VoxelNet[9]	LiDAR	77.47	65.11	57.73
PointPillars [11]	LiDAR	79.05	74.99	***68.30***
PointRCNN[12]	LiDAR	**85.94**	75.76	**68.32**
SECOND[10]	LiDAR	84.04	75.38	67.36
SECOND+$\mathcal{L}_{\text{IoU}}$	LiDAR	84.43	***76.28***	68.22

Table 7. Comparison with other public methods on the KITTI testing dataset for 3D "Car" detection. For easy understanding, we have highlighted the top two numbers in bold and italic for each column. All the numbers are the higher the better.

categories. Especially, the detection rates have been improved by a big margin when we have a higher IoU threshold. Furthermore, for the mAP criterion, both the $\mathbf{L}_{\text{IoU}}$ and $\mathbf{L}_{\text{GIoU}}$ also give a big improvement compared with the original PointRCNN. Based on this experiment, we can conclude that the proposed IoU loss can also work for two-stage based method.

5.4. Comparison with Other Methods

In the above subsections, we have compared the implemented new loss with $\mathbf{L}_1$ loss based on different baselines. In this subsection, We compare the improved baseline with state-of-the-art methods of 3D object detection on both val split and test split of KITTI [3] 3D object detection benchmark. First of all, Tab. 6 gives the comparison results on validation dataset. We have listed nearly all the top results with publications here including: multi-modalities fusion-based [24, 8, 25, 26, 27, 28], one-stage- [10, 9, 11] and two-stage-based [12] approaches. Among all the methods, the improved baseline with $\mathcal{L}_{\text{IoU}}$ and $\mathcal{L}_{\text{GIoU}}$ achieved the best results on all the three categories and it even performs much better than other fusion-based and two-stage-based methods.

Tab. 7 gives the evaluation results on the KITTI testing benchmark. We achieved the results on testing split submitting the results on KITTI's online evaluation server and the results of other methods are obtained from their publications respectively. One important thing is that our model submitted to the test server is trained with half-half split as used on the validation dataset rather than using a bigger training split (*e.g.*, [11]). From the table, we can find that the proposed loss improved the performance of the baseline [10] for all the three types. Especially for "moderate" and "hard" categories, the improvement nearly reaches one point. Furthermore, for the "moderate" and "hard" types, the proposed loss achieved comparable or even better results with the state-of-the-art sensor fusion-based [28] or two-stages-based methods [12].

6. Conclusion and Future Works

In this paper, we have addressed the 2D/3D object detection problem by introducing the IoU loss for two rotated Bboxes. We proposed a unified framework independent IoU loss layer which can be directly applied for axis-aligned or rotated 2D/3D object detection frameworks. By integrating this IoU loss layer into several state-of-the-art 3D object detectors, consistent improvements have been achieved for both 2D detection in bird-eye-view and 3D object detection in point cloud. Especially, the proposed IoU loss performs much better when the IoU threshold is set at a high value. In the future, we would like to extend the current IoU loss layer to more general 3D object detection cases, *e.g.*, Bboxes with three orientation parameters.

References

[1] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520. ACM, 2016. 1, 2, 3, 4

[2] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. *arXiv preprint arXiv:1902.09630*, 2019. 1, 2, 3, 4, 5

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1, 3, 4, 5, 8

[4] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 2, 4

[5] 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. 1, 2, 3

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2, 3, 4

[8] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 1, 2, 8

[9] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 1, 2, 5, 8

[10] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1, 2, 5, 6, 8

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784*, 2018. 1, 2, 5, 7, 8

[12] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 2, 5, 7, 8

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2, 3, 4

[15] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154, 2014. 2

[16] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 2

[17] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016. 2

[18] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 2

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2, 5

[20] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and bounded iou loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6877–6885, 2018. 3, 6

[21] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 3

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 3

[23] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 4

[24] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 8

[25] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 8

[26] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018. 8

[27] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Ipod: Intensive point-based object detector for point cloud. *arXiv preprint arXiv:1812.05276*, 2018. 8

[28] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *arXiv preprint arXiv:1903.01864*, 2019. 8