International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 11 | Nov 2022 www.irjet.net p-ISSN: 2395-0072

Real-time 3D Object Detection on LIDAR Point Cloud using Complex-YOLO V4

Tony Davis[1], Nandana K V[2]

Abstract - *Point cloud-based learning has increasingly attracted the interest of many in the automotive industry for research in 3D data which can provide rich geometric, shape and scale information. It helps direct environmental understanding and can be considered a base building block in prediction and motion planning. The existing algorithms which work on the 3D data to obtain the resultant output are computationally heavy and time-consuming. In this paper, an improved and optimized implementation of the original Complex-YOLO: Real-time 3D Object Detection on Point Clouds is done using YOLO v4 and a comparison of different rotated box IoU losses for faster and accurate object detection is done. Our improved model is showing promising results on the KITTI benchmark with high accuracy.*

***Key Words*: 3D Object Detection, Point Cloud Processing, Lidar, Autonomous Driving, Real-time**

1. INTRODUCTION

With the rapid improvement in 3D acquisition technologies in real-time, we can extract the depth information of the detected vehicle which help to take accurate decisions during autonomous navigation. Point cloud representation encompasses the original geometric 3D spatial data without discretization. Therefore, it is considered the best method for spatial understanding related applications such as autonomous driving and robotics. However, deep learning in point clouds faces several significant challenges [1] such as high dimensionality and the unstructured nature of point clouds. Deep learning on 3D point clouds [2] is a known task in which the main methods used for 3D object detection can be divided into two categories:

1) Region proposal based method

2) Single-shot methods

Region Proposal based methods. These methods first propose several regions in the point cloud to extract region-wise features and give labels to those features. They are further divided into four types: Multi-view based, Segmentation based, Frustum based, and other methods.

- Multi-View based methods fuse different views coming from different sensors such as a front camera, back camera, LiDAR device, etc. These methods have high accuracy but slow inference speed and less real-time efficiency. [3] [4]

- Segmentation based methods use the pre-existing semantic segmentation based methods to eliminate the background points and then to cluster the foreground points for faster object inference. These methods have more precise object recall rates as compared to other Multiview based methods and are highly efficient in highly occluded environments. [5]

- Frustum based methods, first use the 2D object detection based methods to estimate regions for possible objects and then analyse these regions using 3D point cloud based segmentation methods. The efficiency of this method depends on the efficiency of the 2D object detection method. [6]

  Single Shot Methods. These methods directly predict class probabilities using a single-stage network.

- Bird's Eye View (BEV) based methods work on BEV of the point cloud, which is a top-view (2D representation) and use Fully Convolutional Networks to detect bounding boxes. [7]

- Discretization based methods first try to apply discrete transformations on the point clouds and then use Fully Convolutional Networks to predict the bounding boxes. [8]

  The study focus was mainly a trade-off between accuracy and efficiency. The KITTI [9] benchmark is the most used dataset in research. Based on results achieved on the KITTI benchmark on various 3D object detection algorithms, Region proposal based methods outperform single-shot methods by a large margin in KITTI test 3D and BEV Benchmarks [2]. Also regarding autonomous driving, efficiency is much more important for real-time performance. Therefore, the best object detectors are using region proposal networks (RPN) [10] [11] or a similar grid-based RPN approach [12].

  In this paper, an improved and optimized version of the original complex-YOLO [13] is achieved using YOLO v4 [14]. In addition to the implementation of complex-YOLO in YOLO V4, the paper also does the comparison of various rotated box IoU loss towards mean average precision. We can see an overall improvement in performance as well as the efficiency with the optimization.

1.1 Related Work

Before Chen et al. [3] created a network that takes the bird's eye view and front view of the LIDAR point cloud as well as an image as input. It first generates 3D object proposals from a bird's eye view map and projects them to three views. A deep fusion network is used to combine region-wise features obtained by ROI pooling for each view as shown in "Fig. 1". Then the fused features are used to jointly predict object class and perform 3D bounding box regression. They proposed a multi-view sensory-fusion model for 3D object detection in the road scene that takes advantage of both LIDAR point cloud and images by generating 3D proposals and projecting them to multiple views for feature extraction. A region based fusion network is introduced to deeply fuse multi-view information and do oriented 3D box regression. The approach significantly outperforms existing LIDAR based and image-based methods on tasks of 3D localization and 3D detection on the KITTI benchmark [9]. Their 2D box results obtained from 3D detections also show competitive performance compared with the state-of-the-art 2D detection methods. Although this method achieves a recall of 99.1%, its speed is too slow for real-time applications

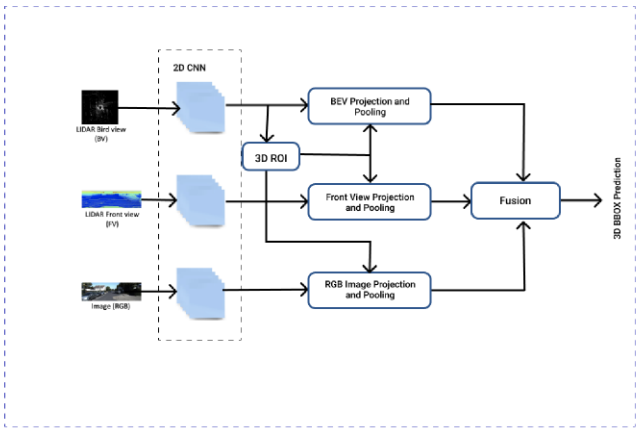

Fig. 1. Multi view RPN based 3D Object Detection by Chen et al. [3]

Zhou et al. [15] proposed a model that operates only on Lidar data. Concerning that, it is the best-ranked model on KITTI for 3D and bird-eye view detections using Lidar data only. The basic idea is end-to-end learning that operates on each grid cell without using handcrafted features. Despite the high accuracy, the model ends up in a low inference time of 4fps on a TitanX GPU [15]. Simon et al. [13] proposed a model that was able to achieve 50 fps on Nvidia Titan X compared to other approaches on the KITTI dataset [9]. He used the multi-view idea (MV3D [3]) for point cloud pre-processing and feature extraction. However, he neglected the multi-view fusion and generate one single birds' eye-view RGB-map that is based on Lidar only, to ensure efficiency. Euler's Region Proposal Network(E-RPN) method is an additional feature by which the author can get the orientation of the objects using imaginary and real parts, this helps exact localization of 3D objects and accurate class predictions as shown in "Fig. 2". Complex-YOLO is a very efficient model that directly operates on Lidar only based BEV RGB-maps to estimate and localize accurate 3D multiclass bounding boxes. The mentioned approach is processed on the LIDAR data in a single forward path which is computationally efficient.

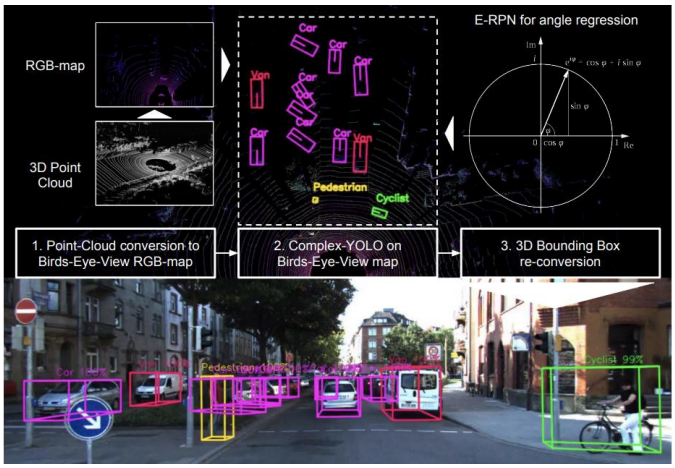

Fig. 2. The point cloud is first preprocessed and converted to RGB(height, intensity and density) map. Then the complex-YOLO model is performed on the BEV RGB map and further optimized with ERPN re-conversion. The lower part shows the projection of 3D boxes to image space.Adapted from [13] by Simon et al.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

2. MODIFIED COMPLEX-YOLO

2.1 Complex-YOLO

The complex-YOLO model by Simon et al, which uses a simplified YOLOv2 [16] CNN architecture which has 18 convolutional and 5 max pool layers, as well as 3 intermediate layers for feature reorganization respectively, extended by a complex angle regression and E-RPN, to detect accurate multiclass oriented 3D objects while still operating in real-time. The 3D point cloud data, which can cover a very large area is confined to a smaller area 80m x 40m to generate bird's eye view RGB map. The R channel refers to Height, G channels to Intensity and B channel to Density of the point data, the size of the grid map used is 1024x512 resolution and a constant height of 3m for all the objects. Calibration data obtained from the KITTI dataset [9] is used to encode the points to the respective grid cell of the RGB map. The maximum height, intensity and density of points in each grid cell is encoded to that grid cell which results in an image that encodes the point cloud information.

The input for complex-YOLO architecture is the encoded BEV image which can be processed as a normal RGB image using yolov2 with the addition of complex angle regression by ERPN method, to detect multi-class parts of 3D objects. The translation from 2D to 3D is done by a predefined height based on each object class. Simon et al. propose an Euler Region Proposal (ERP) which considers the 3D objects position, class probability, objectiveness and orientation. To obtain proper orientation the Simon et al. has modified the normal grid search based approach by adding the complex angle.

The YOLO Network "Fig. 3" divides the image into a grid (16 X 32) and then, for each grid cell, predicts 75 features.

5 boxes per grid cell. YOLO predicts a fixed set of boxes, 5 per grid cell.

- For each box, the box dimensions, and angles [Tx, Ty, Tw, Tl, Tim, Tre], where Tx, Ty, Tw, Tl are the x, y, width, and length of the bounding box. Tim, Tre is the real and imaginary parts of the angle of bounding box orientation. Hence, 6 parameters per bounding box.
- 1 parameter for objectness probability, i.e. probability of the predicted bounding box containing an object and how accurate is the bounding box.
- 3 parameters of probabilities of a bounding box belonging to each class (car, pedestrian, cycle).
- 5 additional parameter alpha, Cx, Cy, Pw, Pl used in the calculations shown in the E-RPN. So, for each grid cell, there are 5 bounding boxes and each bounding box has (6 + 1 + 3 + 5) parameters. That makes it 75 parameters per grid cell.

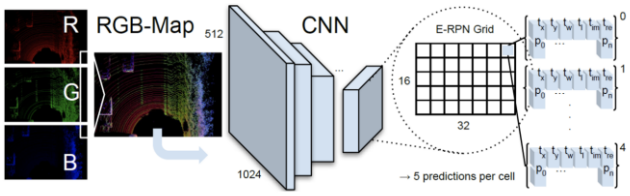

Fig. 3. Complex-YOLO architecture [13] by Simon et al.

The performance evaluation of Complex-YOLO is done with a comparatively older network (YOLO v2 [16]) in the paper. Still, compared to the latest available networks for bounding box detection on 3D point clouds, Complex-YOLO provides a good trade-off between accuracy and inference speed.

2.2 Yolo V4

YOLO v4 [14] claims to have the state of the art accuracy while maintaining a high processing frame rate. In Object detection having high accuracy is not the best performance metrics anymore. These models should be able to run in low cost hardware for a production application. In YOLO v4 [14], improvements can be made in the training process to increase accuracy. Some improvements have no impact on

the inference speed and are called Bag of Freebies while some improvements impact slightly on the inference speed and termed as Bag of Specials. These improvements include data augmentation, cost function, soft labelling, increase in the receptive field, the use of attention etc. The Yolo

1) Backbone: YOLO v4 uses Cross stage partial connections (CSP) with Darknet-53 as the backbone in feature extraction. The CSPDarknet53 [17] model has higher accuracy in object detection compared to ResNET based designs even though the ResNET has higher classification accuracy. But the classification accuracy of CSPDarknet53 can be improved with Mish [18] and other techniques.

2) Neck: Every object detector has a backbone in feature extraction and a head for object detection. To detect an object at different scales, a hierarchical structure is produced with head probing feature maps at different spatial resolutions. To enrich the feature information fed into the head, neighbouring feature maps coming from the bottom up or top downstream are either added or concatenated before feeding into the head. YOLO v4 uses modified spatial pyramid pooling (SPP) "Fig. 4" [19] and modified Path Aggregation Network (PAN) [20]

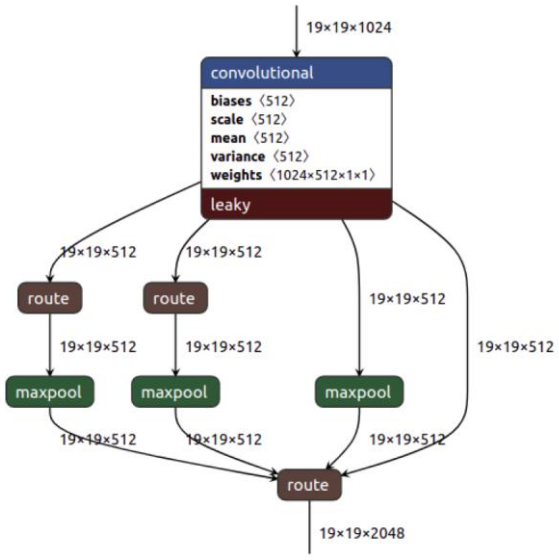

Fig. 4. SPP observed in YOLOv4

3) Head: This is a network that is in charge of actually doing the detection part (classification and regression) of bounding boxes. The head used "Fig. 4" in YOLO v4 is the same as YOLO v3 [21].

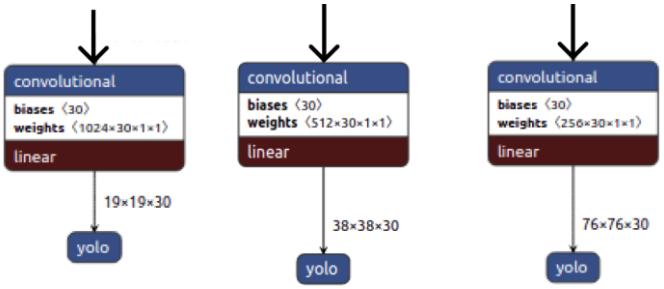

Fig. 5. YOLO Head applied at different scales

2.3 Data Augmentation

Data augmentation increases the generalization ability of the model. To do this we can do photometric distortions like changing the brightness, saturation, contrast and noise or we can do geometric distortion of an image, like rotating it, cropping, etc. These techniques are a clear example of a BoF, and they help the detector accuracy. There are various techniques of augmenting the images like CutOut [22], Random rescale, rotation, DropBlock [23] which randomly masks out square regions of input during training

2.4 Cost Function

To perform regression on coordinates the traditional thing is to apply to mean squared error. But this method is inefficient as it doesn't consider the exact spatial orientation of the object and can cause redundancy. To improve this Intersection over union (IoU) loss was introduced which considers the area of the predicted Bounding Box and the ground truth Bounding Box. However, intersection over union only works when the predicted bounding boxes overlap with the ground truth bounding box. It won't be effective for non-overlapping cases. The idea of IoU was improved using Generalized intersection over union (GIoU) [24] to maximize the overlap area of the ground truth bounding box and the predicted bounding box. It increases the size of predicted BBox to overlap the ground truth BBox by moving slowly towards the ground truth BBox for non-overlapping cases. It takes several iterations to achieve this. Another bounding box regression method used is Distance intersection over union (DIoU) [25] which directly minimises the distance between predicted and target boxes and converges much faster.

3. TRAINING & EXPERIMENTS

The original Complex-YOLO [13] model which used YOLO v2 is evaluated by improving the model by changing the network to newer YOLO v4 using KITTI Dataset [9]. The Velodyne point cloud data from KITTI is used as input to the Complex-YOLO model after preprocessing the data to create the required birds-eye view RGB map. The training labels of the object dataset is used as input labels. Camera calibration matrices of object dataset and left colour images are used for creating the visualization of predictions.

3. 1 Point cloud Prepossessing

The first step in the training pipeline is to convert the 3D lidar point clouds into BEV. Just as in images three channels are used in creating the BEV, the RGB map of the point cloud is composed of height, intensity and density. First, we will decide the area to encode as the lidar point cloud is a large spatial data. So, we will set boundaries for point cloud based on the front side of the vehicle to the reference mounting point of the sensor. We also fix the width and height of the RGB map in this case -25m - 25m across the y-axis and 0m to 50m across the x-axis. We also discretize the feature map based on the maximum boundary along the x-axis and height of the BEV map to create grids. After separating the search space into grids, we can then encode each grid cell for its height, intensity, and density based on the number of points contained in each grid cell. To encode height and intensity the maximum value of points inside that grid cell is used. The density of the grid cell is calculated as "(1)" [13]

$$zr = \min(1; \log(count + 1) / \log 64) \quad (1)$$

The final result after generating BEV map is shown in "Fig. 6"

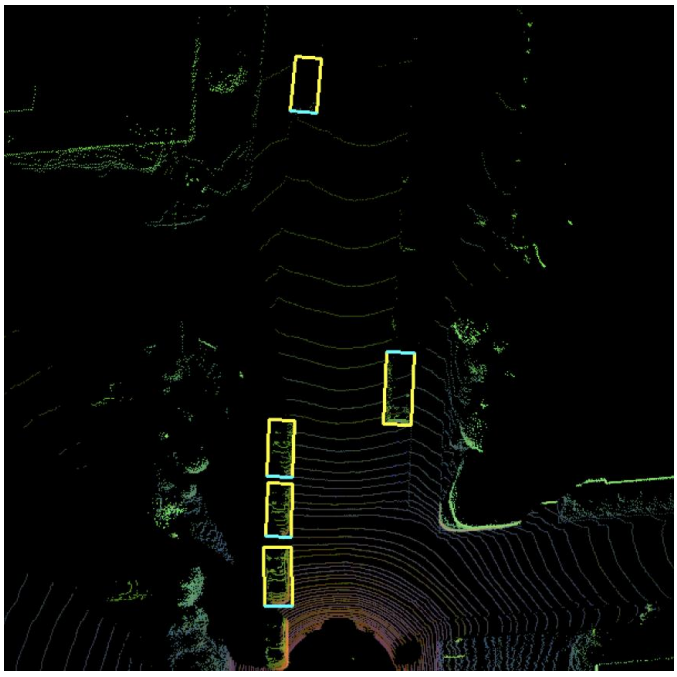

Fig. 6. Labeled BEV RGB map

3. 2 Training

As mentioned in the Complex-YOLO [13] the momentum and decay were set to 0.94 and 0.0005 respectively. The input size: 608 x 608 x 3. The training set contains 7481 training lidar data that were divided into validation (1481) and training set (6000). Adam [26] optimizer is used to train

instead of SGD mentioned as per the main paper. An epoch of 150 is used as the aspect ratio of the image is high but we needed an epoch of about 300 as our model does inductive learning and for proper generalization from the training data to fit any data we needed higher epochs. The learning rate is increased gradually based on the epoch. Mish [18] activation and batch normalization are also used between the layers.

3. 3 Evaluation

For evaluation, the IoU threshold and non-max suppression threshold is set at 0.5 for all the 3 classes. Average precision is taken as the metric for evaluation.

3.4 Rotated IoU Losses

The performance of the model was compared using three different bounding box regression algorithms,3d, IoU, GIoU [24] and DIoU [25]. Bounding box regression was done using the IoU threshold of 0.5 and found the best 3d bounding box and then applied ERPN [13] to find the orientation and direction of the bounding box.

Fig. 7. Model prediction in KITTI [9] test data. The upper part is re projection of the 3D boxes into image space

3.5 Results

The model showed high performance in NVIDIA RTX 2060 GPU. The orientation of the 3D objects was also successfully determined. The comparison of the performance under different rotated IoU losses is listed in Table I. Also, in "Fig. 7" shows the model's prediction on test data where the model can detect highly occluded objects to the camera image.

TABLE I
PERFORMANCE COMPARISON FOR 3D OBJECT DETECTION

Method	*Car(AP)*	*Pedestrian(AP)*	*Cyclist(AP)*	*mAP*	*FPS*
IoU loss	0.92	0.52	0.75	0.73	22
GIoU loss	0.93	0.52	0.67	0.71	14
DIoU loss	0.94	0.60	0.82	0.79	24

4. CONCLUSION

In this paper an updated and optimized Complex-YOLO for a real-time efficient model for 3D object detection using LIDAR only data is done using YOLO v4. Different Bag of freebies and Bag of specials were added to increase the accuracy with no effect or limited effect on inference speed. It's observable that with much less epoch and without proper generalization the evaluation results show a good overall average precision in all three classes. Also, the comparison of different rotated bounding box losses was done and concluded that

1) IoU loss would not provide any moving gradient for nonoverlapping cases and fails when predicted and ground truth boxes don't overlap. Also, the convergence speed of IoU loss is slow.

2) In general, GIoU loss achieves better precision than IoU loss. In this case, as the aspect ratio of the input image is high GIoU loss gives inaccurate regression and slow inference time. It solves the vanishing gradients for nonoverlapping cases.

3) DIoU loss converges much faster than IoU and GIoU loss and has improved accuracy compared to others.

As our model uses a single forward path, The inference speed is high compared to other multi fusion models for 3D object detection for real-world application using less powerful GPUs. This model can be improved further using Complete IoU loss (CIoU) for bounding box regression and other bag of freebies and bag of specials.

REFERENCES

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. (2017) Pointnet: Deep learning on point sets for 3d classification and segmentation. [Online]. Available: https://arxiv.org/abs/1612.00593

[2] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. (2020) Deep learning for 3d point clouds: A survey. [Online]. Available: https://ieeexplore.ieee.org/document/9127813
[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. (2017) Multi-view 3d object detection network for autonomous driving. [Online]. Available: https://ieeexplore.ieee.org/document/8100174

[4] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun. (2018) Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. [Online]. Available: https://ieeexplore.ieee.org/document/8403277

[5] S. Shi, X. Wang, and H. Li. (2019) Pointrcnn: 3d object proposal generation and detection from point cloud. [Online]. Available: https://ieeexplore.ieee.org/document/8954080

[6] D. Xu, D. Anguelov, and A. Jain. (2018) Pointfusion: Deep sensor fusion for 3d bounding box estimation. [Online]. Available: https://ieeexplore.ieee.org/document/8578131
[7] B. Yang, W. Luo, and R. Urtasun. (2018) Pixor: Real-time 3d object detection from point clouds. [Online]. Available: https://ieeexplore.ieee.org/document/8578896

[8] B. Li, T. Zhang, and T. Xia. (2016) Vehicle detection from 3d lidar using fully convolutional network. [Online]. Available: https://arxiv.org/abs/1608.07916

[9] A. Geiger, P. Lenz, and R. Urtasun. (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. [Online]. Available: https://ieeexplore.ieee.org/document/6248074

[10] R. Girshick. (2015) Fast r-cnn. [Online]. Available: https://arxiv.org/abs/1504.08083

[11] S. Ren, K. He, R. Girshick, and J. Sun. (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. [Online]. Available: https://arxiv.org/abs/1506.01497

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. (2016) You only look once: Unified, real-time object detection. [Online]. Available: https://ieeexplore.ieee.org/document/7780460

[13] M. Simon, S. Milz, K. Amende, and H.-M. Gross. (2018) Complexyolo: An euler-region-proposal for real-time 3d object detection on point clouds. [Online]. Available: https://arxiv.org/abs/1803.06199

[14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. (2020) Yolov4: Optimal speed and accuracy of object detection. [Online]. Available: https://arxiv.org/abs/2004.10934

[15] Y. Zhou and O. Tuzel. (2018) Voxelnet: End-to-end learning for point cloud based 3d object detection. [Online]. Available: https://ieeexplore.ieee.org/document/8578570

[16] J. Redmon and A. Farhadi. (2016) Yolo9000: Better, faster, stronger. [Online]. Available: https://arxiv.org/abs/1612.08242

[17] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh. (2019) Cspnet: A new backbone that can enhance learning capability of cnn. [Online]. Available: https://arxiv.org/abs/1911.11929

[18] D. Misra. (2019) Mish: A self regularized non-monotonic activation function. [Online]. Available: https://arxiv.org/abs/1908.08681

[19] K. He, X. Zhang, S. Ren, and J. Sun. (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. [Online]. Available: https://arxiv.org/abs/1406.4729

[20] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. (2018) Path aggregation network for instance segmentation. [Online]. Available: https://arxiv.org/abs/1803.01534

[21] J. Redmon and A. Farhadi. (2018) Yolov3: An incremental improvement. [Online]. Available: https://arxiv.org/abs/1804.02767

[22] T. DeVries and G. W. Taylor. (2017) Improved regularization of convolutional neural networks with cutout. [Online]. Available: https://arxiv.org/abs/1708.04552

[23] G. Ghiasi, T.-Y. Lin, and Q. V. Le. (2018) Dropblock: A regularization method for convolutional networks. [Online]. Available: https://arxiv.org/abs/1810.12890

[24] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. (2019) Generalized intersection over union: A metric and a loss for bounding box regression. [Online]. Available: https://arxiv.org/abs/1902.09630

[25] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren. (2019) Distanceiou loss: Faster and better learning for bounding box regression. [Online]. Available: https://arxiv.org/abs/1911.08287 [26] D. P, Kingma, and J. Ba. (2017) Adam: A method for stochastic optimization. [Online]. Available: https://arxiv.org/abs/1412.6980