Real-Time Object Recognition for Football Field Landmark Detection Based on Deep Neural Networks

Muhamad Rezki Dwijayanto
Electrical Engineering Department
Politeknik Negeri Batam
Batam, Indonesia
Rezki.smkn5@gmail.com

Sumantri Kurniawan
Electrical Engineering Department
Politeknik Negeri Batam
Batam, Indonesia
sumantri@polibatam.ac.id

Budi Sugandi
Electrical Engineering Department
Politeknik Negeri Batam
Batam, Indonesia
budi_sugandi@polibatam.ac.id

***Abstract*—The main goal of this study is to detect the landmark of football fields with a real-time object recognition system based on deep neural networks. The deep neural networks used are YOLO (*You Only Look Once).* The deep learning of YOLO can detect and recognize objects robustly in a high speed. By using this method, the robot can robustly detect and recognize the landmark of football fields in images with size of 608 x 608 pixels at 16 frames per second with MAP (Mean Average Precision) and IoU (Intersect over Union) with 97.60% and 81.36% accuracy.**

***Keywords*—Detection Landmark of Football Fields, YOLO, Real-Time**

I. INTRODUCTION

There are many aspects need to be improved in robotic system. One of the important parts need to be improved is on the localization of positioning robot, especially for the wheeled robot. The robot must know his position before executing the command. Because when the robot doesn't know his position, executing the command can make the robot crashed or hurt the human.

The wheeled robot is used in many sectors in industry, like warehouse robot. Warehouse robot is a robot that moves items around a factory floor or warehouse. Because so many industries is using wheeled robot for his sector, there are so many contest which aims to improve the wheeled robot system. Indonesian is part of country that creates the contest that using wheeled robot.

The Contest is one of the divisions of the Indonesian Robot Contest. This contest is played with two teams of robots based on the rules performed in the Middle Size League (MSL) RoboCup. Each team consists of 3 robots (1 keeper and 2 attackers). The goal of this match is the victory achieved by each team based on the number of goals. The POLIBATAM team has participated in the Indonesian Wheeled Robot Contest from 2017. To win this competition, the POLIBATAM team needs robotic optimization and strategy.

This capability requires each robot to know the robot's position information on the field. The problem with discussing an autonomous mobile robot is the problem of navigation, positioning problems, and mapping problems. Determining the position is defined as the process of determining the local position robot to the environment or the map given. The Important part from determining the position is to detection the landmark of the football fields. The landmark of football fields is used to make maps for robots. In the football fields on the RoboCup contest. There are four landmarks can use to create maps of the football fields, there are "O", "L", "T", and "X" like shown in Fig. 1.

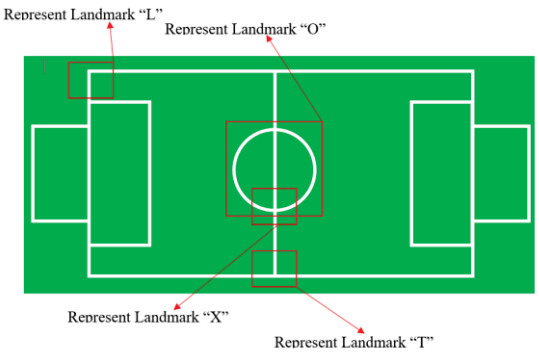

Fig. 1. Represent The football fields on the RoboCup.

II. OBJECT RECOGNITION USING YOLO

You Only Look Once (YOLO) [1], is a computer vision system that can facilitate one image similar to RetinaNet [2], but with superior inference speed when compared to a system developed using SSD [3], R - FCN [4] and FPN FRCN [5]. Its speed makes it one of the most suitable systems for real-time object detection. One of the main features of the problem is, where the model is to complete the bounding box (BB), along with the problem of certain areas that might contain objects. Unlike the other systems, where proposals or regions are generated as candidate objects, where classes inference are executed. This tolerance provides YOLO benefits in terms of speed, it must be processed every image just to do some detection, from the proposal process individually. Also, it draws conclusions from the global image template, making it smaller helping it to discipline the background template, which translates into positive numbers. discuss with other systems.

Another distinctive feature is the end-to-end training process, which means having unique pipes that are trained together. Unlike other systems that have different components and need to be trained separately, such as Faster RCNN [6]. Since the release of the YOLO release,

978-1-7281-2807-8/19/$31.00 ©2019 IEEE

there are have 3 main versions of the algorithm, each for increasing the accuracy of the previous version. The version, YOLOv1 [1], reached 63.4% mean average precision (mAP) above PASCAL VOC 2007 [7], with 45 FPS inference speeds. Introduction of fully convolutional models [3], multi-scale training [8], batch normalization [9], prior BB dimensions [8], among other techniques, lift YOLOv2 [8], which gets a map of 48.1% on the COCO dataset [10] and 78.6% maps on PASCAL VOC 2007, while working at 40 FPS.

The latest version of YOLO, YOLOv3 [11], includes a larger model with 75 convolutional layers that use block residues [12], BB prediction on 3 different scales using procedures similar to pyramid network features [5], among other enhancements which produces maps of 57.9% on COCO, at 20 FPS on TitanX GPUs that are the same as all models in which conversations. The difference between the YOLO version and the one that clearly illustrates the performance that can be sacrificed to get the speed of setting can choose the most suitable version for the application given. Besides, there are versions of YOLOv1 and YOLOv2 that are even faster. Reviews of most YOLO versions are calculated in Table 1.

TABLE I. PERFORMANCE OF VARIOUS YOLO VERSIONS.

Model	Input Size	Train Set	Test Set	mAP	FPS
YOLOv1	448×448	VOC 2007 + 2012	VOC 2007	63.40%	45
Fast YOLOv1	448x448	VOC 2007 + 2012	VOC 2007	52.70%	155
YOLOV2	416x416	VOC 2007 + 2012	VOC 2007	76.80%	67
tiny-YOLOv2	416x416	VOC 2007 + 2012	VOC 2007	57.10%	207
YOLOv2	608×608	COCO	COCO	48.10%	40
YOLOv3	608×608	COCO	COCO	57.90%	20

III. DETECTION LANDMARK USING YOLO

To implement YOLO we use Asus X555-LB. The specification of this notebook is Core I7-5500U with NVIDIA Geforce 940M, and also we use Logitech C920 as the camera for capturing the image. In general process YOLO capturing an image from the camera, for capturing the image. The image capture image is processed by the CPU. Then the image which we get from the camera will be processed on GPU. After the process complete the image is passing to CPU again for displaying the image. The block diagram of the general process of YOLO is shown in Fig. 2.

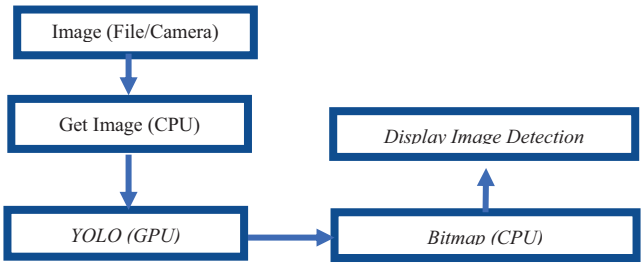

Fig. 2. Block diagram proces YOLO

The architectures of YOLO offer residual skip connections and up-sampling. The most prominent feature of v3 is making detection on three different scales. YOLO is a fully convolutional network and is finally generated by using kernel 1 x 1 on the feature map. In YOLO v3, detection is done by applying 1 x 1 kernel detection on map features with three different sizes in three different places in the network.

Kernel detection of YOLO v3 have a construction 1 x 1 x (B x (5 + C)). B as the number of bounding boxes can be predicted by cells in the map feature, 5 is getting from 4 bounding box attributes and one object trust, and number of classes is represented as C. In YOLO v3 that using COCO datasheet has the construction B = 3 and C = 80, so the kernel detection is 1 x 1 x 255. This kernel has the same height and width then generated the feature map, and it has detection attributes along the levels.

This work we used the fastest version of YOLOv3, it's called YOLOv3-tiny. The quiet difference from normal YOLOv3 it's in the total layer that used, YOLOv3-tiny is used 24 layers for the system detection. It means we process for detection object is less than YOLOv3 with 106 layers [13].This work is training YOLOv3-Tiny with its datasheet. We called the datasheet is a landmark datasheet. On this datasheet we use 4 classes there are landmark "X", "T", "L" and "O". The total of an image on the datasheet is 125 data represent all classes.

TABLE II. LAYERING YOLOV3-TINY FOR SIZE 608 × 608 THAT USED ON THIS WORK.

Layer	Filters	Size	Input	Output
0 Conv	16	3 × 3 / 1	608 × 608 × 3	608 × 608 × 16
1 Max		2 × 2 / 2	608 × 608 × 16	304 × 304 × 16
2 Conv	32	3 × 3 / 1	304 × 304 × 16	304 × 304 × 32
3 Max		2 × 2 / 2	304 × 304 × 32	152 × 152 × 32
4 Conv	64	3 × 3 / 1	152 × 152 × 32	152 × 152 × 64
5 Max		2 × 2 / 2	152 × 152 × 64	76 × 76 × 64
6 Conv	128	3 × 3 / 1	76 × 76 × 64	76 × 76 × 128
7 Max		2 × 2 / 2	76 × 76 × 128	38 × 38 × 128
8 Conv	256	3 × 3 / 1	38 × 38 × 128	38 × 38 × 256
9 Max		2 × 2 / 2	38 × 38 × 256	19 × 19 × 256
10 Conv	512	3 × 3 / 1	19 × 19 × 256	19 × 19 × 512
11 Max		2 × 2 / 2	19 × 19 × 512	19 × 19 × 512
12 Conv	1024	3 × 3 / 1	19 × 19 × 512	19 × 19 x1024
13 Conv	256	1 × 1 / 1	19 × 19 x1024	19 × 19 × 256
14 Conv	512	3 × 3 / 1	19 × 19 × 256	19 × 19 × 512
15 Conv	27	1 × 1 / 1	19 × 19 × 512	19 × 19 × 27
16 Yolo				
17 Route 13				
18 Conv	128	1 × 1 / 1	19 × 19 × 256	19 × 19 × 128
19 Up Sample		2 ×	19 × 19 × 128	38 × 38 × 128
20 Route 19	8			
21 Conv	256	3 × 3 / 1	38 × 38 × 384	38 × 38 × 256
22 Conv	27	1 × 1 / 1	38 × 38 × 256	38 × 38 × 27
23 Yolo				

The configuration of 416 × 416 pixels and 608 × 608 pixels are the same, the difference is only in the size of the object. The general configuration, YOLO uses convolution filter 3 × 3 and 2 × 2 Max-pooling as used in the VGG model. Following the YOLO Nine (Network in Network) work system uses a global average collection to make predictions as well as a 1 × 1 filter for 3 × 3 compression convection. YOLO uses batch normalization to stabilize training, accelerate convergence, and set models [8].

Following the YOLO9000 system a tiny-YOLOv3 system predicts bounding boxes using dimension clusters as anchor boxes. Anchor boxes initialize size width or height, which is closest to the size of the object, then it will be changed to the size of the object using the output neural network (final feature map). This network predicts 4 coordinates in each bounding box, t_x, t_y, t_h, t_w [8]. If the cell is in the upper left corner of the image based on (C_x, C_y) and the bounding box has the width and height P_ (w,) P_h, then the prediction becomes:

$$b_x = \sigma(t_x) + C_x \quad (1)$$

$$b_y = \sigma(t_y) + C_y \quad (2)$$

$$b_w = P_w e^{tw} \quad (3)$$

$$b_h = P_h e^{th} \quad (4)$$

The formula at 1, 2, 3 and 4 is the formula used to predict the bounding box at YOLOv2 and YOLOv3 [8, 11]. During the YOLO training process using the sum of squared error loss. If the basis of truth for some coordinates predicts ˆ t *, then the gradient is the basic truth value (calculated from the ground truth box) minus prediction: ˆ t * - t *. At YOLOv3 predict the objectness score using logistic regression. This is worth one (100%) if the previous bounding box overlaps with the ground truth object more than the other bounding boxes. If the bounding box was not the best before but overlaps with a ground truth of more than a few stresses then the prediction is ignored [11]. The stress used in YOLOv3 is 0.5.

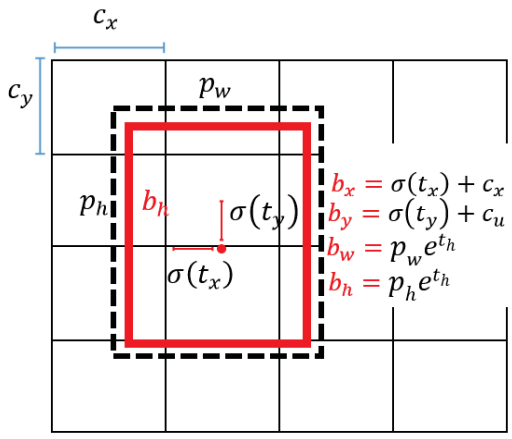

Fig. 3. Represent the formula Bounding [9].

The training process used the pre-trained model of the darknet, darnet53.conv74, which means a pre-trained model that has been trained with 74 convolutional layers followed by average pool-layer with validation accuracy from The imagenet single crop top-5 of 93.8%. Then the model is converted to performance detection and a convolutional layer is added. In the training process using the linear activation function for the last layer and another layer using the leaky rectified activation function, namely :

$$\phi(x) = \begin{cases} x & if x > 0 \\ 0.1x, & Otherwise \end{cases} \quad (5)$$

Then the sum-squared error in optimization is output from the model.

$$SSE = \sum (y - y')^2 \quad (6)$$

Sum-squared errors represent error weight in large boxes and small squares. YOLO predicts many bounding boxes, but YOLO sets the bounding box based on the largest IOU (Intersect over Union). IoU is how much it overlaps on detection boxes and learning boxes [10].

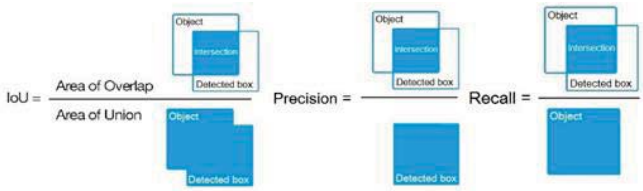

Fig. 4. Represent the formula of IoU, Precision, and Recall.

Each prediction will be more confident in predicting the size, aspect ratios, or class of objects, and increasing each recall and precession. Recall is a true positive preposition of output, while precession is a true positive preparation from the detection box.

In the training process optimization of multi-function is [8]:

$$\begin{aligned} &\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y - y')^2 \right] + \\ &\lambda_{coord} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \\ &\sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \\ &\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \\ &\sum_{i=0}^{S^2} 1_{ij}^{obj} \sum eeclasses \left(P_i - \hat{P}_i \right)^2 \end{aligned} \quad (7)$$

The above formula is used to localize, look for confidence boxes and do localization as in appendix XII. To test learning results used Average Precision (AP). Average Precision is calculated based on an average of the maximum precision of 11 levels of recall.

$$AP = \frac{1}{11}\sum r\varepsilon\{0.0,\ldots,1.0\} AP_r = \frac{1}{11}\sum r\varepsilon\{0.0,\ldots,1.0\} \text{Pinterp}^r \quad (8)$$

The AP used here is the AP at PASCAL VOC 2007 which is true positive if IoU> 0.5 (50%). The results of the AP will be used as MAP (Mean Average Precision). MAP is the average of the AP of each object class [7].

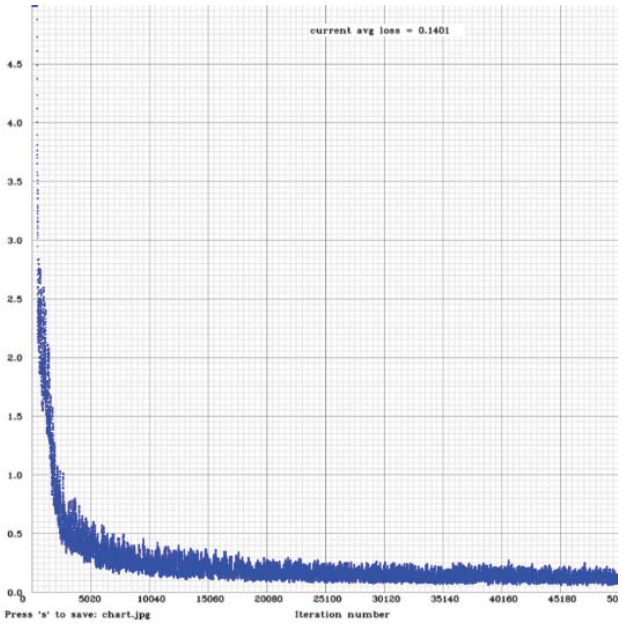

Fig. 5. Chart of the lowering error while learning the datasheet.

From the chart above, we can see that the reduction of error is very good and fast. Besides of reduction error very fast, the average loss error also has a good value that is 0.0006. This better very good than when we used PASCAL VOC and COCO datasheet. This caused on landmark datasheet the object is very different to each other. As the result, the different object will increase the effectivity of learning object.

The learning process is carried out as many as 50200 iterations, the iteration assignment processing up to the maximum point of error is 20000 iterations. In the next iteration, there is an increase in error at the same point. This same increase and decrease occurring approximately 30000 iterations, and the data in the graph is taken in every 100 iterations.

Our system will automatically store weight so that when the learning process is complete the program produces 502 weight but the weight that is at the lowest error point is approximately 300 weight. Then 300 weight is done by calculating MAP and IoU on each weight. MAP and IoU calculations use personal datasheet collected from the Indonesian Wheeled Robot Contest division field photos on the Indonesian Wheeled Robot Contest division POLIBATAM team, which has 125 test data objects. The data contains landmark image data in the form of "L", "T", "X" and "O". MAP and IoU calculations are done automatically using the program, after completing the calculation, the program saves the data into the text file. Then the text file is selected manually using Microsoft Word to select the highest MAP and IoU data using the Microsoft Word find feature. This process takes less than 1 day, and the data obtained is as follows:

TABLE III. : THE RESULT OF THE AVERAGE PRECISION OF THE YOLOV3-TINY ON USING LANDMARK DATASHEET.

Class	**Average Precision**	
	416 × 416/FPS= 30±	*608 × 608/FPS= 16±*
L	81.67%	100%
T	100%	100%
X	79.03%	90.39%
O	99.84%	100%
Mean	90.13%	97.60%
IoU	83.94%	81.36%

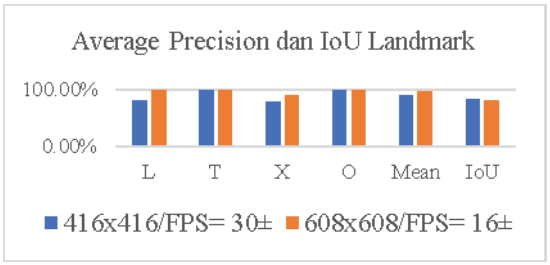

Fig. 6. The chart of Average precision and IoU of Landmark.

The data in Table 3 and Fig. 4 are obtained based on testing of the datasheet by resizing 416 × 416 and 608 × 608, then detecting and matching between the datasheet and the detection results so that we can see the detection of 4 class objects having very good precision values because they have mAP (mean Average Precision) 90.03% and 97.60%. This result is better than detection with 20 class objects and 4 class objects before, in addition to classes on objects only 4 datasheets used for learning have the same size so that when resized there is no difference between object classes. In this test, all objects can be detected very well.

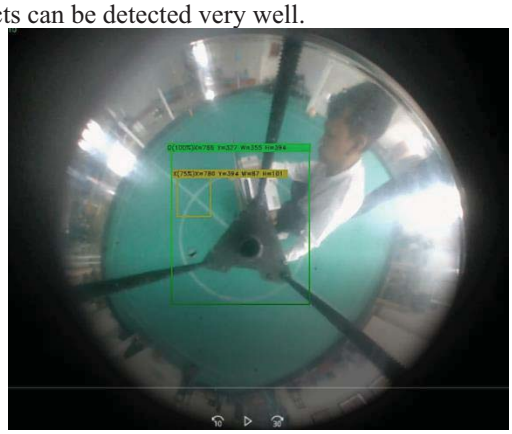

Fig. 7. Result the YOLOv3-Tiny using video record.

On the figure above that we can see, the result of this work is very good the percentage of the detection object is 100% ± when detected class "O" then the percentage of the detection object is 75% ± when detected class "X", we conclude the size of object "X" is too small then other, because on the other object is above 90% ± with size bigger than classes "X".

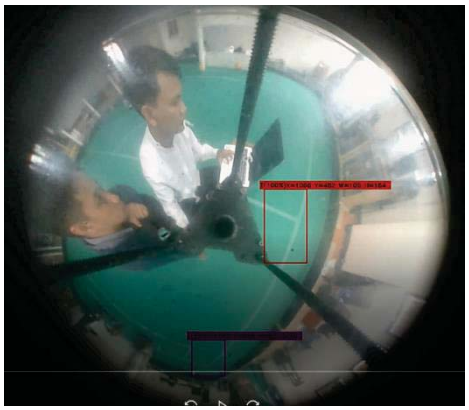

Fig. 8. The picture of tested YOLOv3-Tiny using video Record.

When detected object class "T" and class "L" the percentage of the detection object is 100% ± and 99% ±.

IV. Conclusions and Future Work

In this work, we studied the use of YOLO for detection landmarks of the Indonesian Wheeled Robot Contest. We Tested on the YOLOv3-Tiny Model. We concluded that YOLO uses 3 main processes to detect objects, namely changing the size of the frame that will be detected to the size of 416 × 416 or 608 × 608, then using convolutional networks to detect objects, generating boxes that have the object classification results after detection results are unified into 1 box by using Non-Maximum Suppression. The Non-Maximum Suppression procedure starts by selecting the best box and assuming that there is indeed an object. Then, the box that is close to the selected box is considered as part of the object and put together into a box that shows the object [9]. Then the result of the datasheet landmark with 4 kind objects that very different shape each other This work tested on the Asus A555L Laptop with VGA 2GB Nvidia GeForce 940 M with Cuda 9.0 and cudnn 7.0. The robot is able to robustly detect and recognize the landmark of football fields in images with resizing 608 x 608 pixels at about 16 frames per second with MAP and IoU 97.60% and 81.36%,this better than when resizing to 416 × 416 with the result 30 frames per second with MAP and IoU 90.03%% and 83.94%.

As future work, we propose to try detection ball and goal area of the Indonesian Wheeled Robot Contest, because this is an important part of the strategy of the Indonesian Wheeled Robot Contest. Then you can try to combine with the positioning robot methods.

References

[1]. Redmon Joseph, et al., "You only look once: Unified, real-time object detection," Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[2]. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Doll´ar, "Focal loss for dense object detection," arXiv preprint arXiv:1708.02002, 2017.

[3]. 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.

[4]. Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, "R-fcn: Object detection via regionbased fully convolutional networks," In Advances in neural information processing systems, pages 379–387, 2016.

[5]. Tsung-Yi Lin, Piotr Doll´ar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," In CVPR, volume 1, page 4, 2017.

[6]. 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.

[7]. M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual objectclasse schallenge: Aretrospective," International Journal of Computer Vision, 111(1):98–136, Jan. 2015.

[8]. Redmon Joseph, A.Farhadi, "Yolo9000: Better faster stronger," Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

[9]. Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.

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

[11]. Redmon Joseph, A.Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv: 1804.02767, 2018.

[12]. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016

[13]. Huang Rachel, Pedoeem Jonathan, Chen Cuixian, "YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers," arXiv preprint arXiv: 1811.05588v1, 14 November 2018.