Convolutional Neural Network for Person and Car Detection using YOLO Framework

M. H. Putra, Z. M. Yussof, K. C. Lim, S. I. Salim
Centre for Telecommunication Research and Innovation (CeTRI),
Faculty of Electronics and Computer Engineering (FKEKK),
Universiti Teknikal Malaysia Melaka (UTeM), Durian Tunggal, Melaka, Malaysia
zulkalnain@utem.edu.my

***Abstract*— In this paper we present a real-time person and car detection system suitable for use in Intelligent Car or Advanced Driver Assistance System (ADAS). The system is based on modified YOLO which uses 7 convolutional neural network layers. The grid cells of the system are varied to evaluate its effectiveness and ability in detecting small size persons and cars in real world images. The experimental results demonstrate that even with 7 convolutional layers, the system is able to provide good detection accuracy and real time operation. Although the mAP scores show reduction in accuracy, the visual qualitative evaluation using real world images indicate the 7 layer YOLO with 11x11 grid cells can correctly and easily detects small size persons and cars. This makes the reduced complexity YOLO a suitable candidate for use in ADAS which demands both relatively good detection accuracy and real time operation.**

***Index Terms*— ADAS; CNN; mAP; YOLO.**

I. Introduction

Vision-based object detection and dimension measurement [1,2] is a hot reseach topic among the computer vision research community. In particular, the person and vehicle detection have a direct application in Advanced Driver Assistance System (ADAS), Intelligent Vehicle and Visual Surveillance System. Various methods have been proposed for person, car detection or general object detection, however majority of the techniques focus on achieving high detection accuracy at the expense of high computational complexity. Hence many of these methods are not suitable for real time applications such as ADAS.

Before the emergence of convolutional neural network (CNN), deformable part model (DPM) [3] using handcrafted features such as HOG [4] has been the state-of-the art object detector for many years. Inspired by the impressive performance demonstrated on image classification, CNN has been applied to object detection and achieves impressive results [5-6]. Most notably, Girshick *et al.* [7] proposed the regions with convolutional neural network (R-CNN) framework for object detection and demonstrated state-of-the-art performance on standard detection benchmarks (e.g., PASCAL VOC [11,12]) with a large margin over the previous arts, which are mostly DPM based. R-CNN uses handcrafted Selective Search algorithm to generate object proposals and CNN classifier for detection tasks. The R-CNN is however computationally expensive due to the forward pass computation required for each proposal. Girshick [8] then proposed Fast-RCNN which reduces computational complexity by sharing convolutional features and pooling object proposals from the last convolutional layer.

While Fast-CNN achieves excellent detection accuracy, its speed is still limited by the bottleneck due to the object proposal generation. A faster version called Faster R-CNN [9] was later proposed which replaces the Selective Search by a region proposal network (RPN) which uses convolutional feature maps to generate object proposals. This allows the object proposal generator to share full-image convolutional features with the detection network, hence enabling the system to achieve further speed-up.

Although Faster-RCNN achieves excellent object detection accuracy, it is computationally intensive and not suitable for use in real time application such as ADAS. To meet the combined requirement of high object detection accuracy and real time operation, a different approach of CNN-based object detection named YOLO was proposed by Redmon *et al*. [10]. In contrast to region proposal-based object detection algorithm such as faster-RCNN, YOLO CNN-based algorithm predicts bounding boxes and class probabilities directly from full images in a single evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. The YOLO model runs in real-time at 45 frames per second on nVidia Titan X with mean average precision (mAP) of 63.4% on the PASCAL VOC 2007 dataset. The fast YOLO achieves a mAP of 52.7% at 150 frame per second (fps) while Faster R-CNN runs at 7 fps and attains a mAP of 73.2% on the VOC 2007 test set

Our person and car detection system is based on modified YOLO architecture, where the number of convolutional layers and classes has been reduced to 7 and 2 respectively. This will result in some reductions in computational complexity but accuracy is expected to degrade. We investigate the performance of the modified YOLO especially for detection of small size person and cars by varying the grid cells from 7x7 to 11x11.

The remainder of this paper is organized as follows. Section 2 provides a brief description of the YOLO architecture. Section 3 briefly describes the datasets used and how training is performed. Section 4 describes experimental results using the system. Section 5 provides the conclusion of this paper.

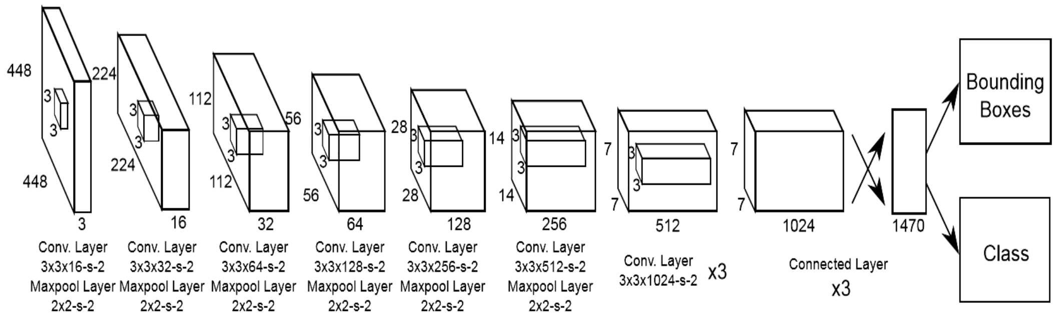
448
448
3
3
224
224
3
3
112
112
3
3
56
56
3
3
28
28
3
3
14
14
3
3
7
7
3
3
7
7
3
16
32
64
128
256
512
1024
1470
Conv. Layer 3x3x16-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x32-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x128-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x256-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x512-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x1024-s-2 x3
Connected Layer x3
Bounding Boxes
Class

Figure 1: Original YOLO architecture [10]

II. You Only Look Once (YOLO)

The YOLO (You Only Look Once) architecture is made up of 27 CNN layers, with 24 convolutional layers, followed by 2 Fully Connected layers and a final detection layer as shown in Figure 1. It divides input image into S × S grid cells and within each grid cell predicts B bounding boxes and a score for each of the C classes. Each bounding box consists of 5 predictions which are center x, center y, width, height and confidence of the bounding box. For each grid cell, there will only be one set of class scores C for all bounding boxes in that region. Hence, the output of the YOLO network will be a vector of S× S × (5B + C) numbers for each image. The fully connected layers use the features extracted from the convolutional layers and use the information to predict the probabilities of the object and at the same time for the bounding box constructions. YOLO final detection layer is a regression that maps the output of the last fully connected layer to the final bounding box and class assignments. The original YOLO network is trained on PASCAL VOC 2007 and PASCAL VOC 2012 dataset with 20 classes of objects with a grid size of 7x7. Figure 2 and 3 show how the network divides the image into grid cells and predict bounding boxes and probabilities for each grid cell. Figure 4 shows the final objects after thresholding is applied.

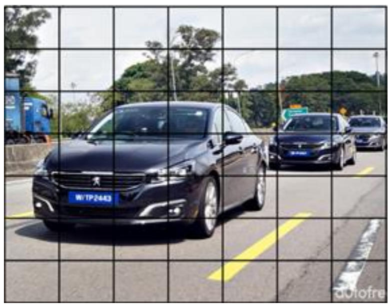

Figure 2: Image is divided into S x S grid cells

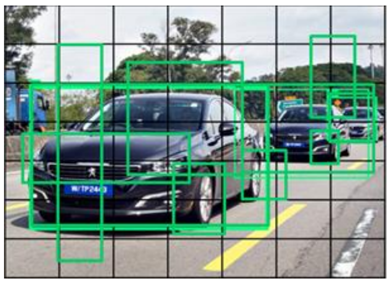

Figure 3: Bounding box is predicted for the image

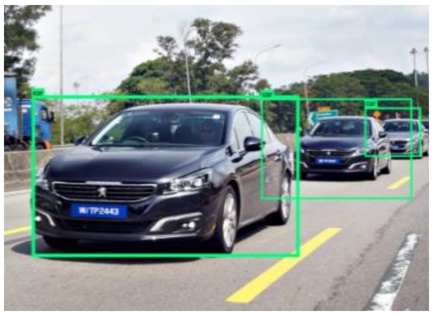

Figure 4: Filtering and filter out best probabilities

III. Experimental Setup

A. The Datasets

INRIA Dataset [4] is used to train our models. In this paper, two classes of object which are person and cars with varying shapes and colours are being used. The resolutions of the images are either 640 x 480 or 480 x 640. The images were generated from different cameras with background of the urban streets. There are 162 training images and 162 test images. In the datasets, the number of the ground truth might be more than one. Usually, the object takes up a small portion of the image and might be distorted from lighting brightness, disorientation and occlusions. Figure 5 shows some of the images in the INRIA datasets used for the experiments.

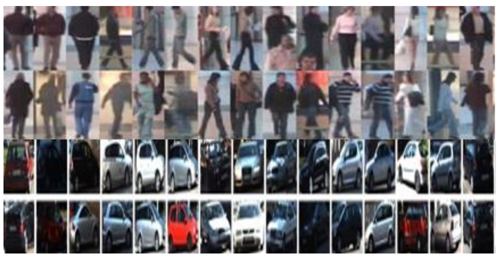

Figure 5: Examples of datasets

B. Training

As for the training, the annotations have been reproduced to indicate the bounding box for the ground truth. The bounding box labelling tool is used for the ground truth coordinate creation. The PYTHON GUI is a modified version of labelling software created by puzzledqs [13] with the interface shown in Figure 6. The labelling tool will generate four points for the coordinate (x1, x2, y1, y2) and class id. The details of the ground truths are saved into a text file and are used during the training. The training for the network is accelerated using Nvidia K40 GPU accelerator which is faster than the normal CPU training speed. After every thousands of iteration, the weight files are stored to a backup directory and can be used as a checkpoint if the training needs to be stopped.

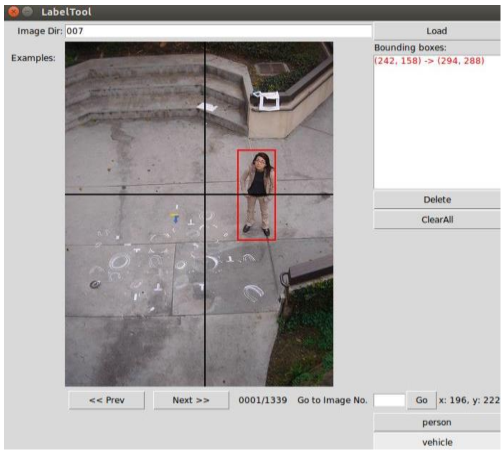
LabelTool

Image Dir: 007

Load

Examples:

Bounding boxes:
(242, 158) -> (294, 288)

Delete

ClearAll

<< Prev

Next >>

0001/1339

Go to Image No.

Go

x: 196, y: 222

person

vehicle

Figure 6: Python GUI for ground truth coordinate labelling

IV. Results

Adopting the YOLO architecture, the parameter of the second last layer has been reduced from twenty classes to only two classes. The last layer of the convolution is connected with the fully connected layer and the final outputs for prediction are varied from 7x7(2 x 5 +2) tensor, 9 x 9(2 x 5 +2) tensor and 11 x 11(2 x 5 + 2) tensor. The prediction from each tensor is compared in terms of accuracy and speed performance.

To evaluate the quantitative performance of the person and car detection system, the system is tested with test images from the INRIA dataset. The test results are provided in Table 1. The system is also tested with images taken from the dashboard camera to evaluate the qualitative performance of the system. Figure 7, 8 and 9 show that the system is capable to detect both cars and persons successfully in different backgrounds.

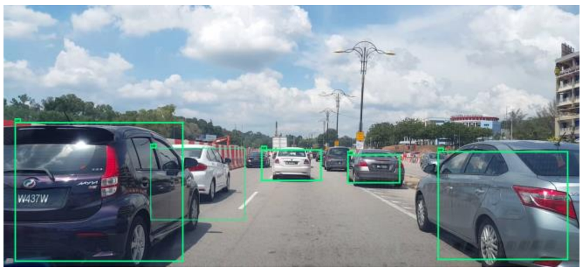
Figure 7: Car detection Result (11 x11 grid cells)

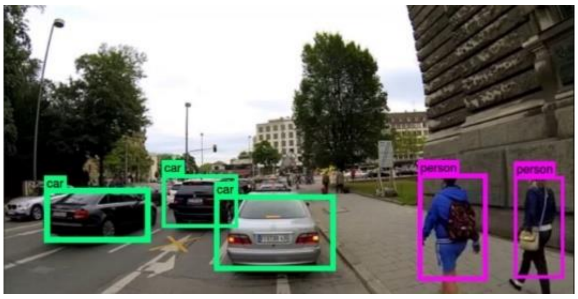

Figure 8: Car and Person Detection Result (11x11 grid cells)

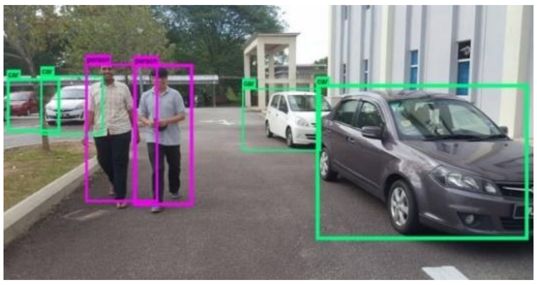

Figure 9: Car and Person Detection Result (11x11 grid cells)

Shown in Figures 10-12 are the test results from the weights produced with grid cells of 7x7, 9x9 and 11x11 respectively. Each of the weights produced is able to locate the person and cars in the images. It can be observed that the accuracy of the object detection in each image increases with the larger number of grid cells. Figure 12 clearly demonstrates that by using 11x11 grid cell allows the system to detect small size car, which is not detected by the other systems that use 7x7 and 9x9 grid cells (see Figure 10 and 11).

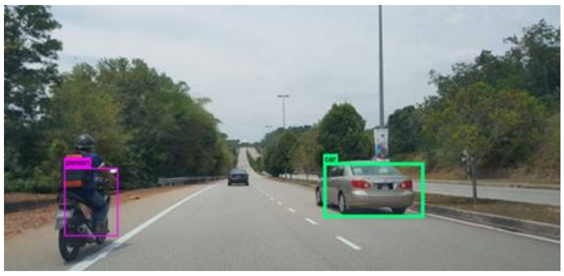

Figure 10: Detection Result with 7x7 grid cells

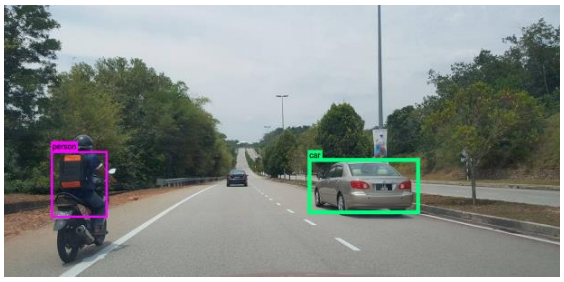

Figure 11: Detection Result with 9x9 grid cells

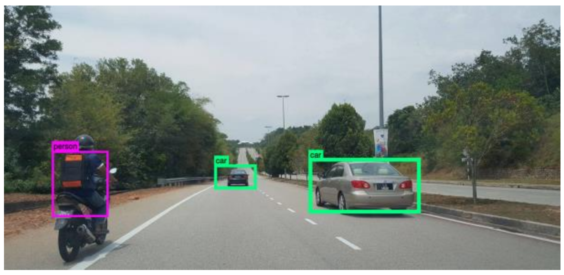

Figure 12: Detection result with 11x11 grid cells

The intersection over union (IOU) accuracy during training is recorded for checking purposes. It is important to observe the trend of the IOU accuracy in order to ensure that YOLO is being trained properly. From the analysis shown in Figure 13, the accuracy level of the IOU for 11 x 11 grid cells implementation gradually increases from 0 to 0.92 for 40,000 iterations.

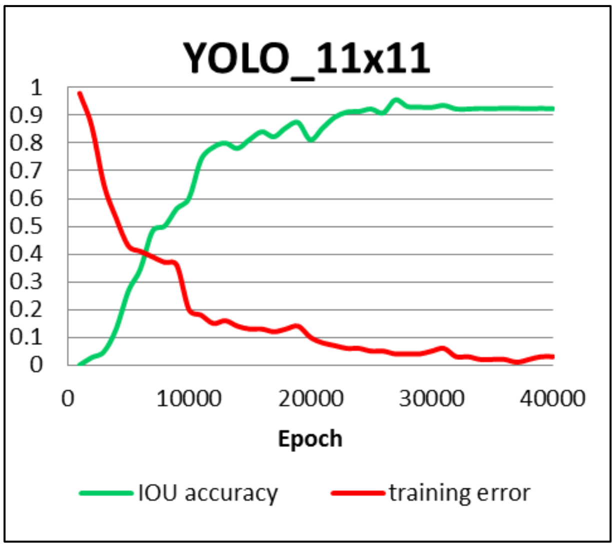

Figure 13: IOU accuracy and training loss

Precision and accuracy of the system is also tested using several methods. The tools and methodology of Hoiem [14] is utilized for each class testing to look at the top-ranked false positives. Each prediction is classified as correct or fall under certain types of error according to the following constraints.

- Correct: correct class and IOU > 0.5
- Localization: correct class, 0.1 < IOU < 0.5
- Similar: class is similar, IOU > 0.1
- Other: class is wrong, IOU > 0.1
- Background: IOU < 0.1 for any object

Figures 14 and Figure 15 show the division of error type across the two classes tested using modified YOLO model utilizing 11 x 11 grid cells alongside with the original YOLO model using 7 x 7 grid cells. The correct classification of the modified YOLO model is rated at 54% which is slightly lower than the original YOLO model. This reduction in accuracy is expected due to the lower number of convolutional layers used in the modified YOLO model.

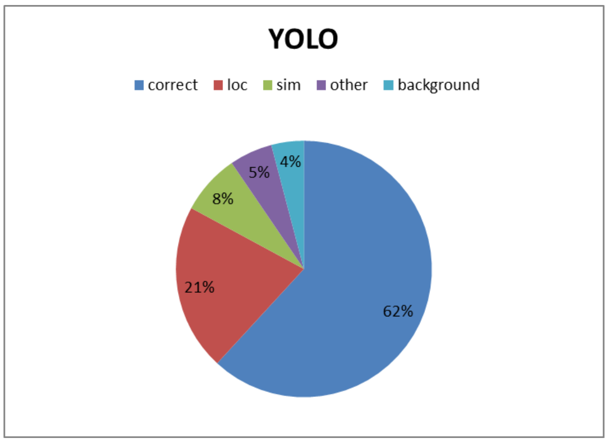

Figure 14: Error Analysis This chart show the percentage of localization and background errors of YOLO.

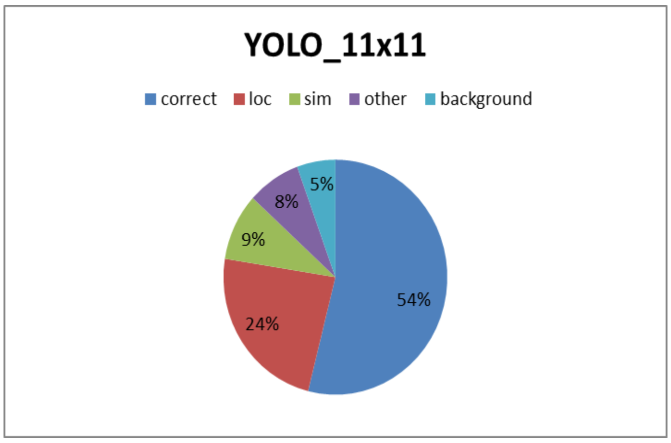

Figure 15: Error Analysis This chart show the percentage of localization and background errors of YOLO_11x11.

The average mean average precision (mAP) is the integral over the precision *p(r)* .

$$mAP = \int_0^1 p(r)dr \tag{1}$$

And the precision is represented by:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{N} \tag{2}$$

where *TP* is the True Positives,
FP is the False Positives and
N is equal to the total number of objects retrieved (*TP* + *FP*).

Table 1 shows the calculated average precision for the original YOLO, YOLO_7x7, YOLO_9x9, and YOLO_11x11. The real-time speed performance in fps is also provided for all of the tested networks running on Nvidia GTX970 GPU.

Table 1
mAP, frame per second and AP performance

Architecture	mAP	Person (AP)	Car (AP)	Frame per Second(fps)
YOLO	59.2	63.1	55.3	26
YOLO_7 x 7	37.9	42.3	33.5	35
YOLO_9 x 9	39.6	43.8	35.3	32
YOLO_11x11	41.1	44.1	38.2	30

From Table 1, it is observed that the detection accuracy improves as the number of the grid cells increases. Using larger grid cells allows the modified YOLO to improve on the detection of small objects. The mAP of YOLO_11x11 is higher than YOLO_7x7 and YOLO_9x9 by 1.5% and 3.2% respectively. YOLO_11x11 with mAP of 41.1% is lower compared to the YOLO with mAP of 59.2% by 18.3%. Since the limitation of original YOLO framework is to detect the small objects, changing the parameter of the grid cell is believed to be a good alternative in improving the accuracy.

As for the speed performance (fps), the size of the grid cells is affecting the speed performance as it requires longer time to process larger grid cells. As for YOLO_7x7 network, it produced the highest speed; however it possesses the lowest precision compared to other network. Therefore, the best trade-off between speed and precision is achieved by YOLO_11x11 network with 30 fps and 41.1 mAP. The higher fps with better precision would enable the modified YOLO model to be integrated in the real-time implementation of ADAS system.

V. CONCLUSION

In this paper we have presented a CNN-based person and car detector with the focus on achieving highest possible detection speeds without significantly sacrificing on detection quality. Our real-time detector is based on modified YOLO which uses 7 convolutional layers. This reduction of number of layers has the impact of reducing the computational complexity at the expense of acceptable loss in detection accuracy. The experimental results demonstrate that although the convolutional layers have been reduced to 7 layers, using larger 11x11 grid cells (or higher) can improve the detection accuracy on small objects. This makes the reduced number of convolutional layers in YOLO with higher number of grid cells a good candidate for use in ADAS which demands both relatively high detection accuracy and real time operation.

ACKNOWLEDGMENT

The authors would like to thank Centre for Telecommunication Research and Innovation (CeTRI), Faculty of Electronics and Computer Engineering (FKEKK), Universiti Teknikal Malaysia Melaka (UTeM), and the MyBrain15 program from the Ministry of Higher Education (MoHE) and Government of Malaysia for sponsoring this study. The authors also would like to thank Centre for Research and Innovation Management Universiti Teknikal Malaysia Melaka (CRIM-UTeM) for sponsoring this paper. Apart from that, the authors also would like to thanks Collaborative Research in Engineering, Science and Technology (CREST) Malaysia for sponsoring the related work and equipments under the research grant (GLUAR/CREST/2015/FKEKK/I00005).

REFERENCES

[1] C. Chen, A. Seff, A. Kornhauser and J. Xiao, DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving Proceedings of 15th IEEE International Conference on Computer Vision (ICCV2015)

[2] J.E. Hoo, KC. Lim, Accuracy and Error Study Of Horizontal and Vertical Measurements with Single View Metrology for Road Surveying, ARPN Journal of Engineering and Applied Sciences. 11(12):7872-6, 2016

[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection, In CVPR, 2005.

[5] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In CVPR, 2014.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector, arXiv:1512.02325 [cs.CV], 2015

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[8] R. Girshick. Fast R-CNN, arXiv preprint arXiv: 1504.08083, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks, arXiv preprint arXiv: 1506.01497, 2015.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, arXiv preprint arXiv: 1506.02640, 2015.

[11] M. Everingham, L. Van Gool, C. K. I.Williams, J.Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007

[12] M. Everingham, L. Van Gool, C. K. I.Williams, J.Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results, 2010.

[13] https://github.com/puzzledqs/BBox-Label-Tool

[14] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error object detectors. In Computer Vision–ECCV 2012, pages 340–353. Springer, 2012.