Object Detection On Indianapolis Racing Car Images Using Retinanet

Manjulata Garimella
mchivuku@iu.edu
Luddy School of Informatics, Computing and Engineering
Indiana University, Bloomington, Indiana

ABSTRACT

Object Detection is used almost everywhere these days. The use cases are endless, be it tracking objects, video surveillance, pedestrian detection, anomaly detection, people counting, self-driving cars, or face detection. The list of possible applications goes on. The goal of my project is to explore the performance of object detection models specifically RetinaNet [9] in detecting cars and smoke in the Indianapolis racing car image dataset. The model is evaluated on mean average precision (MAP) score. Two variants of the model were explored and mAP was computed for each of the RetinaNet model. In the first model, Resnet50 [5] was used as a backbone network for feature extraction and, in the second model Resnet152 [5] was used as a backbone network for feature extraction. Difference between the models is the number of layers of the network. Resnet50 has 50 layers while, resnet 152 has 152 layers. Both the models scored very similar in the mAP score for detection of cars.

RetinaNet model with resnet50 backbone network scored **0.0872 mAP** on detection of car and **0.048 mAP** on detection of smoke on the Indianapolis racing car images while, the model with resnet152 backbone network scored **0.0847 mAP** on detection of car and **0.066 mAP** on detection of smoke on the images. RetinaNet model with resnet50 did well on detection of cars while Resnet152 model performed well in detection of smoke on the test images. Since detection of smoke is more a fine grained problem, addition of more layers with Resnet152 provided better detection model. While, the results on car detection indicated that the model overfitted on the training data. In addition, the mAP values for car is found to be relatively lower than when the model was trained only to detect cars from the training data (mAP score of the model was found to be 0.3). The difference in the performance of the model can be attributed to the model getting confused as the model was trying to detect smoke, where there are mostly cars in the smoke frame.

For future work, adding more training data using data augmentation techniques especially for smoke would help with better training of the model to detect smoke leading to less confusion and improved mAP score for both car and smoke classes. In addition, hyper parameter tuning would be another option that can be explored to understand the improvements of model performance.

KEYWORDS

object detection, RetinaNet, pytorch

1 INTRODUCTION

Object detection is one of the machine learning (ML) applications that has garnered increasing attention from the general AI community in recent times. The goal of object detection is finding all objects in the image and drawing so-called bounding boxes around them. Object detection task has several computer vision applications including motion detection, image classification, bio-metrics, self driving cars, robotics etc., and has significantly improved and advanced with the help of deep learning techniques (especially convolutional neural networks (CNNs)). Being an important research area, there has been tremendous research in building and development of object detection models. Although, there are several such models, finding the appropriate model that suits the given task is often a challenge. The motivation for doing this project was primarily an interest in undertaking such a challenging task and applying to the task of object detection of cars and smoke in the Indianapolis racing car images. I specifically investigated this problem in the context of using RetinaNet [6] model to perform object detection task. In addition, I computed training and inference times of the model to evaluate the model performance in the distributed environment by the use of model parallelization.

RetinaNet [6] is a one-stage object detector that solves the problem of extreme foreground-background class imbalance during training. This class imbalance is solved by reshaping the standard cross-entropy loss in a way that it down-weights the loss assigned to well-classified examples. This reduces the impact of easy negatives in total loss and, the final loss mainly comes from a sparse set of hard examples. RetinaNet solves the speed vs accuracy problem of one-stage object detector and, surpasses of all existing state-of-the-art two stage detectors. Exploring RetinaNet with the focal loss in the car and smoke detection problem can be a great idea because it is fast, accurate and is a one-stage detector. Since, the task is novel and, has not been explored before the results, are independent and cannot be compared with any benchmark results.

In the current report, I provided the results of my experiments performed on RetinaNet under two variants - RetinaNet model with resnet50 as backbone network for feature extraction and RetinaNet model with resnet152 as backbone network for feature extraction. The reports describes comparison of two variants using mAP score (mean average precision) and, visual results of the output images with the bounding boxes.

The remainder of the report is organized as follows. Section 2 briefly reviews related work and background, and Section 3 describes the architecture and implementation details of the approach. Section 4 describes experiments and results and, Section 5 contains conclusion and future work.

1.1 Background And Related Work

Object detection is a well studied problem but detection of cars and smoke from Indianapolis racing car images is a new and novel problem that is yet to be explored. The goal of object detection is to determine where objects are located in a given image (i.e. object

localization) and, which category each object belongs to (i.e. object classification), an example evaluation result of the model is shown in the figure: 1. Each object is detected by the model with certain precision level or score. Input to the model is the image containing objects and, the output of the model is the coordinates of the bounding boxes around each of the object in the image along with the label. Hence, the pipeline of traditional object detection models can be divided into three stages: informative region selection, feature selection and classification.

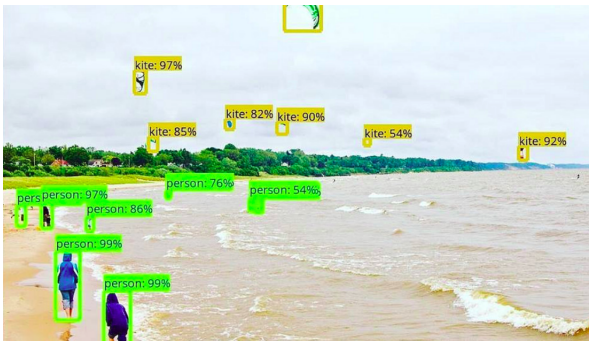

Figure 1: Object detection example

Definitions

- **Image Gradient Vector** - is defined as a metric for every individual pixel, containing pixel color change in both x-axis and y-axis
- **Histogram of Oriented Gradients (HOG)**: The Histogram of Oriented Gradients (HOG) is an efficient way to extract features out of the pixel colors for building an object recognition classifier.
- **Selection Search**: Selective search is a common algorithm to provide region proposals that potentially contain objects. It is built on top of the image segmentation output and use region-based characteristics (not just attributes of a single pixel) to do bottom-up hierarchical grouping.
- **Intersection over union (IoU)**: Is the ratio of intersection (which is the area of the overlap between the bounding boxes) over union (which is the union of area of overlapping bounding boxes, i.e. sum of area of the entire bounding boxes minus the area of the overlap).
- **Evaluation Metrics: mAP** - Mean average precision or mAP is a common evaluation metric used in the object detection tasks. It is a number from 0 to 100; higher value is better. It is computed as follows:
  - Combine all detections from all test images to draw a precision-recall curve (PR curve) for each class. The "average precision" (AP) is the area under the PR curve.
  - Compute AP separately for each of the target classes in the image
  - A detection is a true positive if it has “intersection over union” (IoU) with a ground-truth box greater than some threshold (usually 0.5; if so, the metric is “mAP@0.5”)
- **Non-Maximum Suppression** Non-max suppression helps avoid repeated detection of the same instance. This is done by sorting all bounding boxes by a confidence score and the boxes with low confidence scores are discarded. Greedily select the one with the highest score.
- **RoI Pooling** - It is a type of max pooling to convert features in the projected region of the image of any size, hxw, into a small fixed window, HxW. The input region is divided into HxW grids, approximately every subwindow of size h/Hxw/W and max-pooling is applied on each grid.

The frameworks of object detection methods can mainly be categorized into two types as discussed above: one stage detectors (regression/classification based), two stage detectors (region proposal based). One stage detectors use unified framework to achieve final results (categories and locations) directly. Some examples of these models include YOLO [11], SSD [10], DSSD [1] and DSOD [13] etc. The other class models are the region based models that use two stage approach, matching the attention mechanism of human brain to some extent, i.e. provide a coarse scan of the whole image to extract the regions of interest and, then perform classification on the regions of interest to build final results. Some examples of popular frameworks using this approach are: R-CNN [3], Fast R-CNN [2], Faster R-CNN [12], Mask R-CNN [4] etc. Below is the brief overview on some of these models:

Region Proposal Based Framework

The region proposal based frameworks, are two stage networks, some of the most popular models categorized under these framework are: **R-CNN**: R-CNN [3] adopts to selective search to generate about 2k region proposals for each image. The method then performs a simple bottom-up grouping and saliency cues to provide more accurate candidate bounding boxes of arbitrary sizes quickly thus reducing the search space to bounding box to region candidates ("region of interest" or ROI). Given every image region, one forward propagation through the CNN generates a feature vector. This feature vector is then consumed by a binary SVM trained for each class independently to perform classification. To reduce the localization errors, a regression model is trained to correct the predicted detection window on bounding box correction offset using CNN features. Following are some learning drawbacks of R-CNN models that are considered slow and time-consuming:

(1) Running selective search to propose 2000 region candidates for every image;
(2) Generating the CNN feature vector for every image region (N images *2000)
(3) The whole process involves three models separately without shared computation, CNN for image classification and extraction, SVM for classifier for identifying target objects, regression model for tightening region bounding boxes.

To make R-CNN faster, a new architecture called Fast R-CNN, Faster R-CNN were proposed.

The figure 2 [15] illustrates the design summary of the R-CNN, Fast R-CNN and Faster R-CNN and Mask R-CNN models

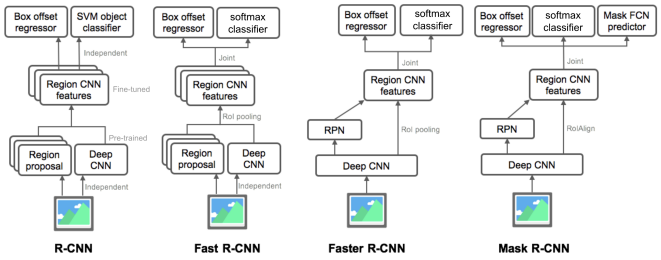

Figure 2: Summary of Models in the R-CNN Family

1.2 Regression/Classification Based Framework

In the region proposal based networks the detection happens in two stages - region proposal generation and feature extraction and classification. The other approach to detection is skipping region proposal stage and run detection directly over the dense sampling of possible locations. These are called one-stage detectors. Some popular frameworks are described below:

(1) **YOLO** [11]: YOLO divides the input image into an SxS grid and each grid cell is responsible for predicting the object centered in that grid cell. Each grid cell predicts B bounding boxes and their corresponding confidence scores. The confidence score indicates the likelihood that the cell contains an object - Pr(containing an object) * IOU(pred, truth); where Pr = probability and IoU = intersection on unions. If the cell contains an objects, it predicts a probability of this object belonging to every class C_i, i = 1.. K: Pr(the object belongs to class C_i | containing an object). At this stage, the model only predicts one set of class probabilities per cell, regardless of number of bounding boxes B. In total, one image contains SxSxB bounding boxes, each box corresponding to 4 location predictions, 1 confidence score, and K conditional probabilities for object classification. The total prediction values for one image is SxSX(5B+K), which is the tensor shape of the final conv layer of the model. The final layer of the pre-trained CNN is modified to output a prediction tensor of size SxSx(5B+K). As a one-stage object detector, YOLO is super fast, but it is not good at recognizing irregularly shaped objects or a group of small objects due to a limited number of bounding box candidates.

(2) **SSD: Single Shot MultiBox Detector**[10]: YOLO has a difficulty in dealing with small objects in groups, which is caused by strong spatial constraints imposed on bounding box predictions. In addition, YOLO struggles to generalize to objects in new/unusual aspect ratios/configurations and produces relatively coarse features due to multiple down sampling operations. Inspired by the shortcomings Single Shot MultiBox Detector (SSD) takes advantage of a set of default anchor boxes with different aspect ratios and scales to discretize the output space of bounding boxes. This can be seen as pyramid representation of images at different scales. In SSD, the detection happens in every pyramidal layer, targeting at objects of various sizes. Unlike YOLO, SSD does not split the image into grids of arbitrary size but predicts offset of predefined anchor boxes (called default boxes) for every location of the feature map. Each box has a fixed size and position relative to its corresponding cell. The network is trained with a weighted sum of localization loss (e.g. Smooth L1) and confidence loss (e.g. Softmax) and final detection results are obtained by conducting NMS on multi-scale refined bounding boxes.

2 ARCHITECTURE AND IMPLEMENTATION

The architecture of the current car and smoke object detection methodology comprises of following components:

- Building annotations for smoke images;
- Preprocessing of input image;
- Data augmentation;
- Network model;

2.1 Building Annotations For Smoke Images

I used labelImg [14] to construct bounding box annotation for the smoke images. I annotated a total of 175 images for training the model to detect smoke.

2.2 Pre-processing

I applied three pre-processing steps on the images. First, normalized images by subtracting the mean RGB value of ImageNet dataset as suggested in [7]. Then, the input pixel range was clipped to 1-0. After that, all the images have been resized to appropriate size of (224,224) to be fed into the network.

2.3 Data Augmentation

Data augmentation is exclusively used to generate more training instances in order to improve generalization ability of the model. For the current project, horizontal flipping of image was used as a data augmentation technique for the project.

2.4 Network Model

The network model that was used for the project is RetinaNet [6].

RetinaNet [8] falls into category of one-stage dense object detector. The architecture of RetinaNet is described in the figure 4. Two crucial building blocks are featurized image pyramid and the use of focal loss.

- **Focal Loss** - The author claimed that the extreme foreground-background class imbalance encountered during training of dense detectors is the central cause for the low accuracy of one-stage detector models. Hence, in order to overcome the problem, the author proposes the use of new loss function that is reshaped over standard cross entropy loss so that the detector will put more focus on hard misclassified examples (i.e. background with noisy texture or partial object) and, to down-weight easy examples (i.e. obviously empty background). Starting with a normal cross entropy loss for binary classification:

$$CE(p, y) = -y \log p - (1 - y) \log 1 - p$$

where y ∈ [0,1] is a ground truth binary label, indicating whether a bounding box contains a object, and p ∈ [0,1] is

the predicted probability of objectiveness (aka confidence score). The equation of focal loss is described below:

$$FL(p_t) = -(1-p_t)^{\gamma} log(p_t) \quad (1)$$

where:

$$p_t = \begin{cases} p & \text{if y = 1} \\ 1-p & \text{otherwise, then } CE(p,y) = CE(p_t) = \log p_t \end{cases}$$

and p ∈ [0,1] is the model estimated probability for the class with label y = 1. γ is the focus-sing parameter ∈ [0,5] that smoothly adjusts the rate at which easy examples are down-weighted. Easily classified samples have high values of p_t >> 0.5 when p is very close to 0 (when y=0) or 1 (when y=1), can incur a loss with non-trivial magnitude.

- **Featurized Image Pyramid** - The featurized image pyramid [7] is a backbone network for RetinaNet. Following the same approach by image pyramid in SSD, featurized image pyramids provide a basic vision component for object detection at different scales. The key idea of the feature pyramid is demonstrated in the figure 3, the base structure contains a sequence of pyramid levels, each corresponding to one network stage. One stage contains multiple convolutional layers of the same size and the stage sizes are scaled down by a factor of 2. Two pathways connect conv layers - bottom up pathway - normal feedforward computation and topdown pathway goes in the inverse direction adding coarse but semantically stronger feature maps back into the previous pyramid levels of a larger size via lateral connections. Finally, these two feature maps are merged by element-wise addition. The lateral connections only happen at the last layer in stages, denoted as Ci, and the process continues until the finest (largest) merged feature map is generated. The prediction is made out of every merged map after a 3x3 conv layer, P_i

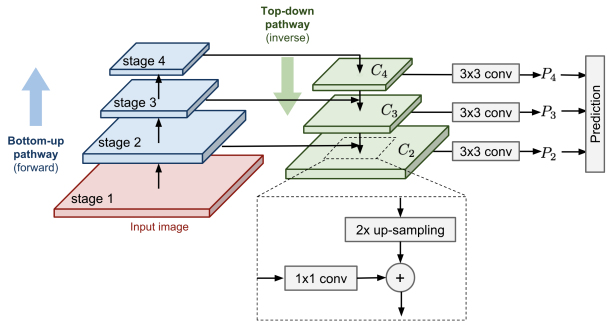

Figure 3: Featurized Image Pyramid Module

The architecture of the model contains 3 components:

- ResNet - used of deep feature extraction
- Feature Pyramid Network (FPN): used on top of ResNet [5] for constructing a rich multi-scale feature pyramid from one single resolution input image. FPN is multiscale, semantically strong at all scales, and fast to compute.
- Two subnets:
  - Classification Subnet - the classification subnet predicts the probability of object presence at each spatial position. The subnet is a FCN which applies four 3×3 conv layers, each with C filters and each followed by ReLU activations,
  - Box Regression Subnet - This subnet is a FCN to each pyramid level for the purpose of regressing the offset from each anchor box to a nearby ground-truth object. It is identical to the classification subnet except that it terminates in 4A linear outputs per spatial location. It is also a class-agnostic bounding box regressor which uses fewer parameters that is found to be equally effective.

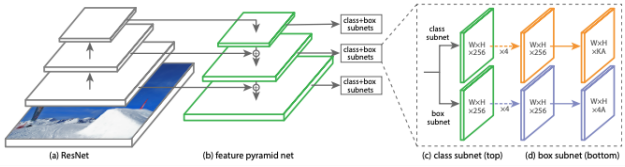

Figure 4: One-stage RetinaNet Architecture

During **Inference** the network decodes box predictions from at most 1k top-scoring predictions per FPN level after thresholding detector confidence at 0.05. Top predictions from all levels are merged and non-maximum suppression (NMS) is applied with threshold of 0.5 to yield final detections. During training total focal loss of an image is computed as the sum of the focal loss over all 100k anchors, normalized by the number of anchors assigned to a ground-truth box.

3 EXPERIMENTS AND RESULTS

3.1 Dataset

(1) Training set contains a total of 2910 images, with ≈ 200 images annotated with bounding box for smoke.
(2) Test set contains a total of ≈ 186 images.

3.2 Training Details

RetinaNet model was trained on two variants:

(1) Resnet-50 as feature extractor - that is pretrained on ImageNet. Adam optimizer was used with learning rate of 1e-5.
(2) Resnet-152 as feature extractor - that is pretrained on ImageNet. Adam optimizer was used with learning rate of 1e-3.

Random horizontal flip was the data augmentation method that was applied. I used a batch size of 2 to train the network. Models were trained on distributed cluster environment (using model parallelization) with multiple GPUs NVIDIA K80. Small batch sized was used to gain the benefit of stochastic gradient descent.

3.3 Testing Details

For test purpose, 2 GPUs were used simultaneously to speedup testing process. I used a default threshold of 0.5:.05:.95 to calculate the average precision. After testing several times, I averaged the values of average precision to compute mean average precision (mAP). Following are the parameters used for evaluation of the model 1

parameter	value	description
IOU threshold	0.5	IOU threshold used to consider when detection is pos or neg
Score threshold	0.05	score confidence threshold to use for detection
Max detections	100	maximum number of detections to use per image

Table 1: Parameter values used for evaluation

Model Name	Average Elapsed Time (sec)
RetinaNet (Model 1)	0.40
RetinaNet (Model 2)	0.66

Table 2: Average Elapsed Time - Result

Model Name	CAR mAP	Smoke mAP
RetinaNet (Model 1)	0.087	0.048
RetinaNet (Model 2)	0.084	0.066

Table 3: mAP Result

3.4 Results:

The table 3 describes the test results of the two models after training the model for 100 epochs. The figure 5 shows training loss for the model 1 with resnet50 as the backbone network for feature extraction and, the figure 6 shows the training loss for the model2 with resnet152 as the backbone network for feature extraction. The figure also contains mAP values for smoke and car that are computed during validation epoch that followed every training epoch. Following are the average inference times for the model 2:

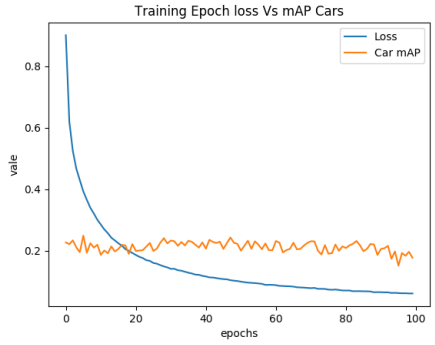

Figure 5: Training loss and mAP over validation set for Model 1

The figure 7 shows visual results from evaluation of object detection on couple of test images.

4 CONCLUSION AND FUTURE WORK

Current approach for object detection using RetinaNet in its two variants can be considered as a baseline result especially for smoke detection. The approach is simple and easy to train and has fast inference times. The mAP score for both the variants of the model were found to be very low in comparison to the model that was trained only to detect car images. The reason for lower performance can be attributed to the confusion caused by the model during detection of smoke in the image where there only cars in the image. In addition, the training data also has a severe class imbalance with more images for car than smoke.

Hence, for future work, following aspects should be considered:

- Add more training data for smoke images with the use of actual images after converting Indianapolis racing car video to frames or by data augmentation techniques;
- Performing hyper parameter tuning to understand the optimal initialization of hyper parameters for better performance;

Finally, I would like to thank Prof. Judy Qui and Selahattin Akkas for the opportunity to work on such an interesting project and, also for providing help along the way for successful completion of the project.

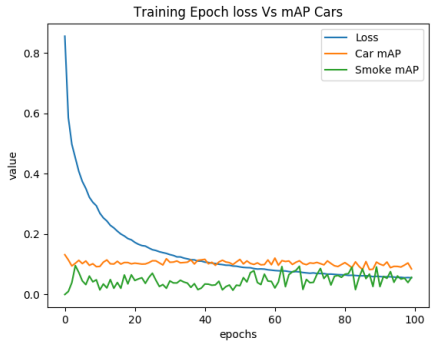

Figure 6: Training loss and mAP over validation set for Model 2

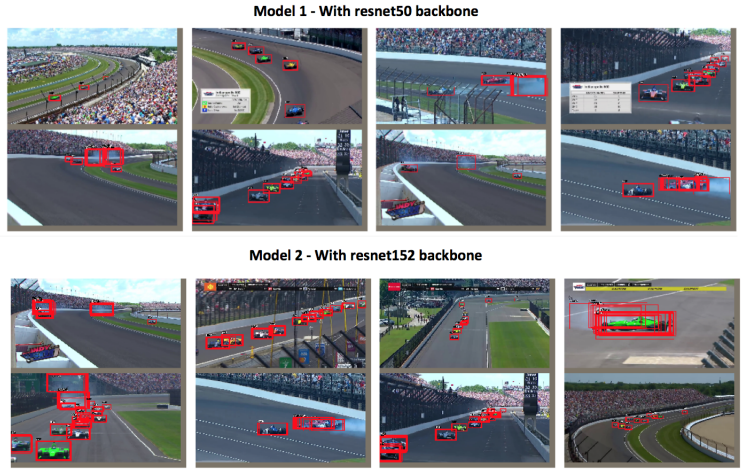

Figure 7: Object detection results: RetinaNet

REFERENCES

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

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

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385* (2015).

[6] Yann Henon. 2018. A Pytorch Implementation of RetinaNet for object detection. (2018). https://github.com/yhenon/pytorch-retinanet

[7] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

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

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

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

[13] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. 2017. Dsod: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE International Conference on Computer Vision*. 1919–1927.

[14] Tzutalin. 2015. LabelImg. https://github.com/tzutalin/labelImg

[15] Lilian Weng. 2017. Object Detection for Dummies Part 3: R-CNN Family. *lilianweng.github.io/lil-log* (2017). http://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html