Vehicles Detection with Faster-RCNN on MobileNetV3

Wong Zhao Wu
School of Computing
Singapore Polytechnic, Singapore
zhaowu.wong@gmail.com

***Abstract*—Vehicle detection is one of the core technologies behind the possibility of autonomous vehicles. Finding the perfect trade-off between accuracy and performance has been a challenge in solving the task of object detection for real-time inference. In this study, my goal is to build an end-to-end light weight object detection algorithm using Faster-RCNN coupled with MobileNetV3 with Feature Pyramid Network (FPN). The final model has obtained mAP of 0.835 on the hold-out validation set.**

Keywords—Object Detection, Faster-RCNN, MobileNetV3, FPN, Vehicles Detection.

I. INTRODUCTION

Computer vision is a subfield of Artificial Intelligence (AI) that enables computers to see and understand the world [1] which is imperative to achieving the ultimate goal of level 5 autonomy of vehicles. While autonomous vehicle is another challenging subject that has seen rapid advancements in the past decades [2], this paper aims to address one of the fundamental challenge of full self-driving, detecting cars in the camera frames which is also known as object detection.

While computer has surpassed human on the top-5 classification error rate on the large scale ImageNet dataset particularly on the task of Image Classification [3], object detection which combines the task of image classification and image localization is another task that receives a lot of research attention.

The state-of-the-art methods can be categorized into two main types: one-stage methods and two stage-methods. While single-stage methods like the YOLO [4] and SSD [5] has been known for real-time inference speed without compromising much on performance, recent advancement of two-stage methods like Faster-RCNN [6] has shown to deliver nearly cost-free region proposals while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012 and MS COCO datasets with only 300 proposals per image.

This paper attempts to perform transfer learning on the Car Object Detection dataset [7] using Faster-RCNN method using ResNet-50 with Feature Pyramid Network as backbone model to generate intermediate and high features for the regional proposal network, to propose set of rectangular object proposals before parsing them to the Fast R-CNN module [8] for classification and bounding box regression.

II. RELATED WORKS

Object detection is an extension of the Image Localization task where by the number of objects in a variable depends on the actual number of object in a frame. A naïve approach to solve this problem would be to take slide a CNN across the image to classify the presence of the object in that region of interest. Such approach, however, raises more problems of strides and aspect ratio selection and a huge toll to inference speed as the frames need to be parsed through a CNN multiple times for a single image, making real-time inference close to impossible.

R-CNN [9] , proposed by Ross et al. address the problem of changing aspect ratio through the introduction of selective search to extract up to two thousands region proposals before resizing and parsing them into a CNN for feature extraction before feeding them into an SVM for classification. Despite the approach able to limits down the number of region proposal, parsing a single image to CNN for two thousands time is still too expensive to be considered for real-time deployment.

Fast-RCNN [8], builds on previous work on proposing regions but instead of proposing it directly from the input image, the region is proposed from the convolutional feature map extracted from a CNN. The primary reason why this approach is considered as "Fast" simply due to the fact that an image only need to parse through a CNN once before further work is done for bounding-box regression and image classification. The bottleneck in this approach is the region proposal method which is commonly implemented on the CPUs and thus, making such runtime inequitable to fast region-based CNNs that takes advantage of GPUs [6].

Faster-RCNN [6] address the bottleneck with the introduction of a Regional Proposal Network which is just another convolutional network, used to generate the region proposal based on predefined anchors with different areas and aspect ratio. Finally, the proposed region of interest is parsed to the same ROI-module from Fast-RCNN table for ROI-Pooling and subsequently classification and bounding-box regression. Such improvement allows faster-rcnn to significantly reduce inference speed to 0.2s per image which makes it a suitable candidate for vehicle detection task.

Other than vehicle detection, several studies take the challenge to another level by introducing vehicle tracking as part of the challenge. In [10] Zhang et al. first trained a YOLOV3 network to obtain the vehicle bounding boxes before calculating the motion similarity from the classic hog feature extraction method. Jia et al. [11], on the other hand, make use of colour as the visual cue to track the movement of vehicle using the Continuously Adaptive Meanshift Algorithm (CamShift). In this paper, since the original dataset does not preserve the temporal information of the cars, I could not incorporate tracking as part of the study scope but it is intriguing to study object tracking with completely different approach proposed.

III. METHADOLOGY

A. Dataset and Image Augmentation

The cars detection dataset obtained from Kaggle [Car Object Detection] has over 1001 annotated image for training and 175 unannotated image for testing. The bounding boxes

for the car label is annotated in the PASCAL VOC format with only a single class available which is "Car". One observation after browsing through the training images is that most of the images are taken from the same camera angles with a similar background of trees and a dark blue sky. This implies that the original dataset, without any augmentation, would only help me to train a model that is only capable of identifying cars on streets with background of trees under a dark blue sky. Another observation is that the training images contains of only cars on roads and thus, changing the horizontal alignment of the image does not affect the semantic context of the vehicles in the image.

Based on the observation gathered, I have implemented several image augmentation policies with the aim of increasing the volume on training image, by synthesizing the augmented version of image, and improving the generalization capabilities of model. The augmentation policies can be classified into two categories. Affine transformations like random horizontal flip, rotation and shearing is applied to make the model more invariance towards translation while colour jittering is applied to simulate the image of cars under different weather and temperature.

Fig. 1 shows some examples of training image after applying the augmentation and its bounding boxes.

Fig. 1. Training Image with Augmentation

B. *Backbone Model*

The first stage of a Faster-RCNN model is the CNN backbone which is used to extract useful features from the images. The original Faster R-CNN paper used VGG-16 [12] as the base network. However, since the goal of this paper was to implement the model for potential real-time inference, VGG-16 is too slow and dense for the purpose. Thus, the MobileNetV3 [13] is a more efficient alternatives in both reducing the computational cost and the number of parameters without compromising much on the accuracy. Table 1 show that MobileNetV3-small is nearly as accurate as VGG-16 while being less computational expensive.

TABLE I. IMAGENET TOP-1 ACCURACY PERFORMANCE

Model	*ImageNet Top-1 Accuracy (%)*	*Multiply-adds (million)*	*Parameters*
MobileNetV3 Small	67.4	56	2.5M
VGG-16	71.5%	15300	138M

Other than the outright speed advantage of MobileNetV3, there are several notable implementations on the model that significantly speed up computation and reduce the size of MobileNet models as compared to other networks. A good example would be the depth-wise separable convolution [13] whereby instead of performing convolution directly with a $3 \times 3 \times d$ convolution filters where 3 is an arbitrary kernel size, the depth wise convolution is achieved through a combination of $1 \times 1 \times d$ expansion convolution, to increase the dimension of input feature map, followed by applying d convolution filters with dimension of $3 \times 3 \times 1$ for each dimension which is also known as the depth-wise convolution. Such implementation significant reduce the amount of multiplication and addition operation while ensuring the output dimension is as intended. Together with another $1 \times 1 \times d$ expansion that ensures the dimension shape matches with the original feature map, the MobileNet is able to implement skip connection just like the one first introduced in the ResNet paper [14] as shown in Fig. 2.

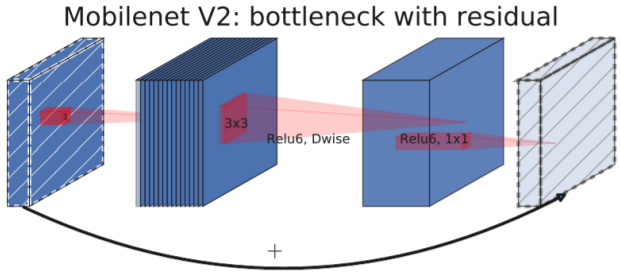

Fig. 2. Inverted Residual Block for MobileNetV2

Moreover, the introduction of Squeeze and Excitation as an attention mechanism in MobileNetV3 also helps in improving the model's performance through providing channel-wise attention with negligible computational cost [15]. The complete inverted residual block with for MobileNetV3 is shown in Fig. 3.

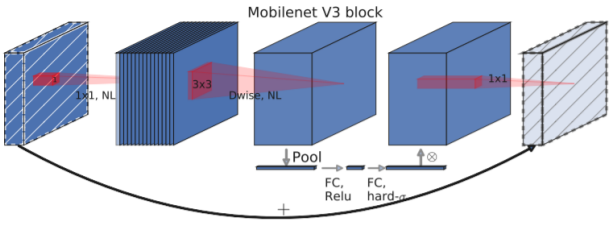

Fig. 3. Inverted Residual Block with Squeeze and Excitation for MobileNetV3

C. *Region Proposal Network (RPN)*

One of the key contribution from the Faster-RCNN paper is the region proposal network to replace traditional region proposing methods with nothing more than just a convolution neural network and predefined anchors boxes. RPN takes in the input feature map from the CNN backbone and output a

sets of initial bounding box values and confidence score for each of the k anchors.

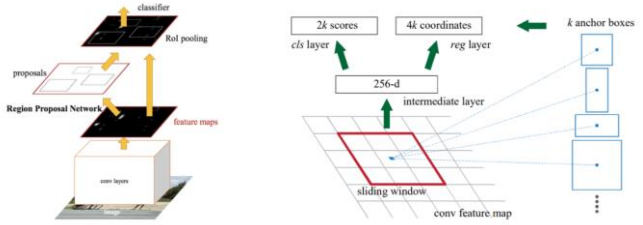

Fig. 4. Regional Proposal Network and Anchors from Faster-RCNN [6]

To be more specific, Fig. 4 illustrates the entirety of the regional proposal network which starts with obtaining the feature map generated by the backbone model. Then, a 3×3 kernel with dimension of 256 is apply for the feature map to further condense the higher level features from the feature map. Since right now the output of the intermediate layer contains very rich spatial and object features, we can then base on every single pixel from the output of the intermediate layer, perform the objectness classification and bounding box regression through two sibling fully connected layers with the first layer having dimensions of $2 * k$ for objectness classification (one value represent object exist in the anchor while the other value represent background class) and another fully connected layer have dimensions of $4 * k$ for initial bounding box regression for each defined anchor.

In this task of vehicle detection, I have realize that the size and aspect ratio of the predefined anchor boxes are a crucial hyperparameter as it facilitates our model to come up with an initial bounding box before parsing the bounding box for further fine-tuned on later part of the network. Hence, I have make use of custom anchor size of $16^2, 32^2, 64^2, 128^2$ and 256^2 pixels to generate smaller anchor such that cars from longer distance can be identified easier than having a large anchor box and find tune from there.

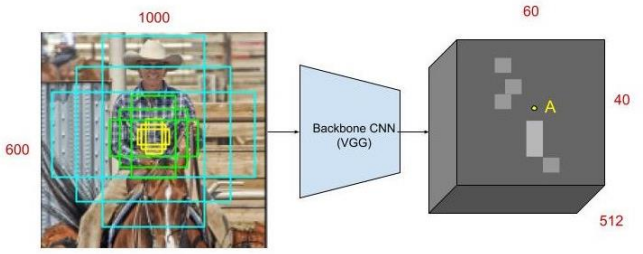

Fig. 5. Possible Anchors in the input image in a location corresponding to point A in the feature map. [16]

D. Feature Pyramid Network

In addition to pure Faster-RCNN detection framework, I have also incorporate the feature pyramid network [17] into my model that can combine lower level feature with higher resolution that is outputted by earlier convolution layer with higher level feature output with lower resolution that is outputted by later part of the convolution layer.

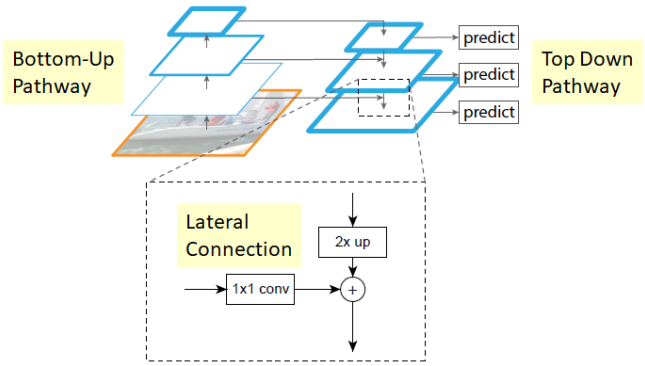

Fig. 6. Feature Pyramid Network for Object Detection [17]

Such top-down pathway and lateral connection [18] shown in Fig. 5 allow us to incorporate earlier features into our regional proposal stage such that they are not missed out in the traditional bottom-up pathway.

FPN also integrates well with the RPN whereby for each of the output feature map by the FPN, the same RPN architecture as described above is implemented to propose regional proposal using anchors of different size and aspect ratio.

E. ROI Pooling and Prediction Head

The final implementation details of Faster-RCNN architecture comes down to the ROI-pooling layer and the prediction head. Since all the proposed region of interest from the RPN have different sizes, and our prediction head is a fully-connected layer that requires a fixed input size, an ROI-pooling layer is used to summaries region proposals of different shape into a $7 \times 7 \times 512$ tensor using max-pooling.

Then, the $7 \times 7 \times 512$ tensor is pass through two layer of FCs before branching of for SoftMax-classification and bounding box regression. Finally, Fig. 7 shows the complete architecture of the Faster-RCNN network modified with MobileNetV3 as the backbone layer and Feature Pyramid Network as the model neck to combine higher and lower-level features for region proposals.

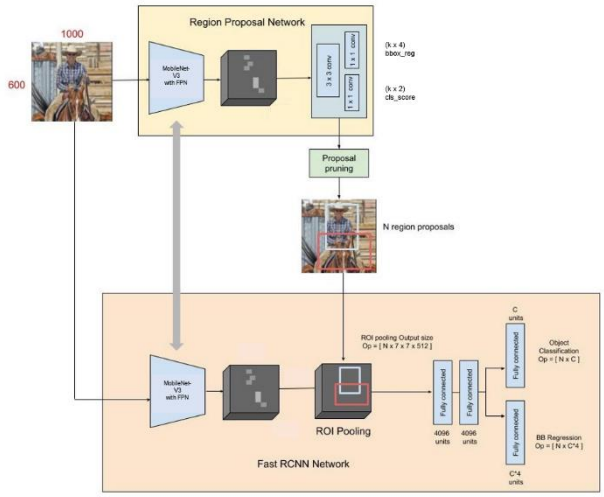

Fig. 7. Complete Faster-RCNN Architechture with MobileNetV3 + FPN

IV. Discussion

A. Training Procedure

Training a Faster-RCNN network is a compound objective minimization task whereby the final loss function that we are minimizing with our optimizer is a sum of multiple sub-loss values including loss_classifier, loss_box_reg from final prediction head and loss_objectness, loss_rpn_box_reg from the RPN.

For the optimizer, I have chosen AdamW [19] with initial learning rate of 5e-4 coupled with the classic step learning rate scheduler that reduce the learning rate with power of 10 for every two epoch. Moreover, I have also applied warm-up scheduler on the first epoch to make the training more stable and allows the model to get a better initial point before larger learning rate kicks in to speed up the process.

B. Training Procedure

For evaluation of the model performance, I have make use of the standard object detection metrics, Average Precision (AP) @ IOU = 0.5. In object detection, a true positive is only counted when the bounding box predicted reflects the actual class and the area of intersection of the predicted bounding box has more than or equal to 50% overlapping with the actual ground truth. The Average Precision(AP) is calculated by finding the area under the precision-recall curve as shown in Fig. 8. Another thing to take note is that the AP curve is often interpolated by replacing each precision value with the maximum precision value to the right of that recall level to smooth out the zig-zag pattern of the AP curve [20]. Model with higher AP means that the model is able to produce more True Positive bounding boxes than False Negative for Recall and False Positive for Precision.

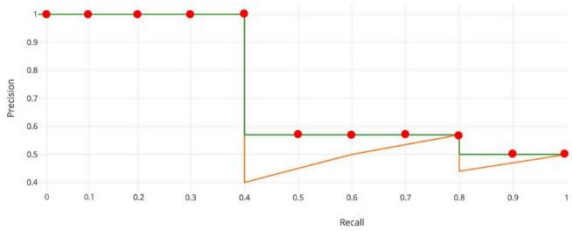

Fig. 8. Recall Precision Curve with Interpolation [20]

For the model created, I manage to obtain AP of 0.835 for the hold-out validation set and AP of 0.929 for the training dataset. This indicates that in some manner, the model suffers from slight overfitting despite the weight decay and heavy data augmentation introduced. The optimal way to solve this issue is to gather more data for both the training set and validation set such that the model can be able to capture more information from the training data and be able to generalize better to different pictures and use cases. Figure [8] shows an example of the prediction of the model before and after the filtering of confidence level at 0.75.

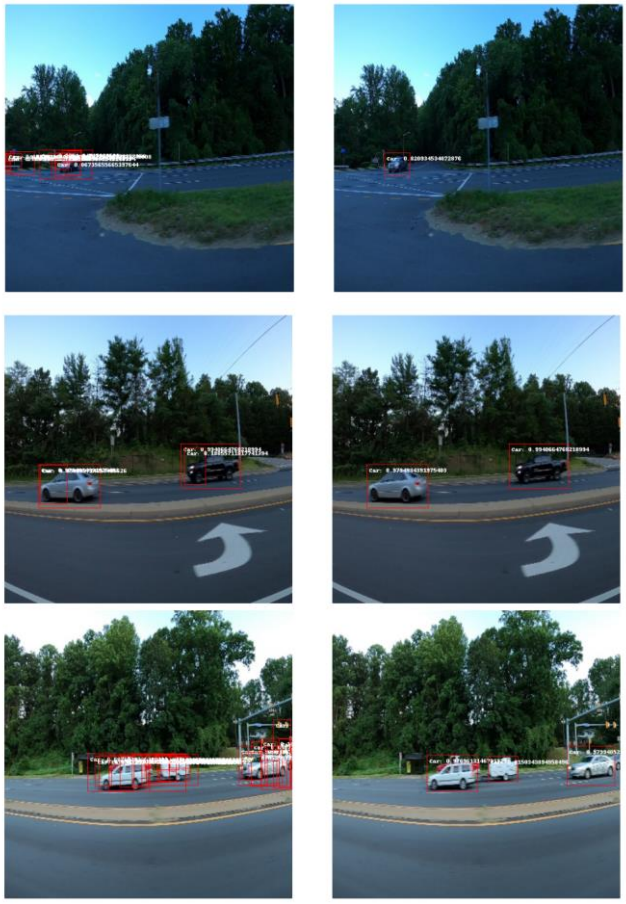

Fig. 9. Visualising Prediction with Before and After Filter of Confidence Level at 0.75

V. Conclusion

FasterRCNN manage to obtain mAP of 0.835 is hold out validation set and by visualising the prediction on validation data, the model is doing quite well for the task of vehicle detection with only 800 annotated image for training after the train-validation split.

There are several notable limitations that is worth mentioning. First, the vehicle detection dataset that I have found is relatively small with only a single classes. Thus performance achieved might simply because the model has overfitted to the validation data without being able to generalize to different cases and cars on the road. Moreover, the current approach of using a two-stage model is still relatively slow and infeasible for deployment despite the attempts of making the model lighter than the one proposed from the Faster-RCNN paper. However, due to the lack of time and computing resource, I did not have the luxury to further source for more comprehensive dataset and explore one-stage model like YOLOs [21] or another popular two-stage model like EfficientDet [22] if I wish to further improve the accuracy.

For this part C of the assignment, I have spent many effort and countless hours attempting to understand the intuition and implementation details of object detection with Faster-RCNN. Finally, I really appreciate this opportunity for me to venture beyond the syllabus and I am looking forward to continue reading and expand my horizon on ways to further improve

the network or even using different SOTA architecture for better performance.

REFERENCES

[1] "What is Computer Vision?" https://www.ibm.com/topics/computer-vision (accessed Nov. 21, 2021).

[2] "Tesla Autopilot," *Wikipedia*. Nov. 13, 2021. Accessed: Nov. 21, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Tesla_Autopilot&oldid=1054984797

[3] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *ArXiv14090575 Cs*, Jan. 2015, Accessed: Nov. 21, 2021. [Online]. Available: http://arxiv.org/abs/1409.0575

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *ArXiv150602640 Cs*, May 2016, Accessed: Nov. 21, 2021. [Online]. Available: http://arxiv.org/abs/1506.02640

[5] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," *ArXiv151202325 Cs*, vol. 9905, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *ArXiv150601497 Cs*, Jan. 2016, Accessed: Nov. 21, 2021. [Online]. Available: http://arxiv.org/abs/1506.01497

[7] "Car Object Detection." https://kaggle.com/sshikamaru/car-object-detection (accessed Nov. 21, 2021).

[8] R. Girshick, "Fast R-CNN," *ArXiv150408083 Cs*, Sep. 2015, Accessed: Nov. 21, 2021. [Online]. Available: http://arxiv.org/abs/1504.08083

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *ArXiv13112524 Cs*, Oct. 2014, Accessed: Nov. 21, 2021. [Online]. Available: http://arxiv.org/abs/1311.2524

[10] Y. Zhang *et al.*, "Research on visual vehicle detection and tracking based on deep learning," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 892, p. 012051, Aug. 2020, doi: 10.1088/1757-899X/892/1/012051.

[11] L. Jia, D. Wu, L. Mei, R. Zhao, W. Wang, and C. Yu, "Real-Time Vehicle Detection and Tracking System in Street Scenarios," in *Communications and Information Processing*, Berlin, Heidelberg, 2012, pp. 592–599. doi: 10.1007/978-3-642-31968-6_70.

[12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv14091556 Cs*, Apr. 2015, Accessed: Nov. 22, 2021. [Online]. Available: http://arxiv.org/abs/1409.1556

[13] A. Howard *et al.*, "Searching for MobileNetV3," *ArXiv190502244 Cs*, Nov. 2019, Accessed: Nov. 22, 2021. [Online]. Available: http://arxiv.org/abs/1905.02244

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *ArXiv151203385 Cs*, Dec. 2015, Accessed: Nov. 23, 2021. [Online]. Available: http://arxiv.org/abs/1512.03385

[15] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks," *ArXiv170901507 Cs*, May 2019, Accessed: Nov. 23, 2021. [Online]. Available: http://arxiv.org/abs/1709.01507

[16] S. Ananth, "Faster R-CNN for object detection," *Medium*, Oct. 01, 2020. https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46 (accessed Nov. 25, 2021).

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *ArXiv161203144 Cs*, Apr. 2017, Accessed: Nov. 23, 2021. [Online]. Available: http://arxiv.org/abs/1612.03144

[18] S.-H. Tsang, "Review: FPN — Feature Pyramid Network (Object Detection)," *Medium*, Mar. 20, 2019. https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610 (accessed Nov. 23, 2021).

[19] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *ArXiv171105101 Cs Math*, Jan. 2019, Accessed: Nov. 25, 2021. [Online]. Available: http://arxiv.org/abs/1711.05101

[20] J. Hui, "mAP (mean Average Precision) for Object Detection," *Medium*, Apr. 03, 2019. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 (accessed Nov. 25, 2021).

[21] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *ArXiv200410934 Cs Eess*, Apr. 2020, Accessed: Nov. 25, 2021. [Online]. Available: http://arxiv.org/abs/2004.10934

[22] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," *ArXiv191109070 Cs Eess*, Jul. 2020, Accessed: Nov. 25, 2021. [Online]. Available: http://arxiv.org/abs/1911.09070