SE MobileNetv2 with Single Shot Detector for Object Detection in Mobile Devices

Anubhav Bhatti
Queen's University
Kingston, Ontario, Canada
anubhav.bhatti@queensu.ca

Abstract

Object detection is one of the essential tasks in the field of computer vision. In today's era, object detection is used in a wide range of applications like robot vision, video surveillance, self-driving cars, etc. With the growing demand for object detection in various real-time applications, it has become essential to develop object detection algorithms with fast processing. To overcome this issue, in this project, I propose a lightweight object detection framework that consists of a modified MobileNetv2 network for learning the feature maps that are further used by an SSD300 pipeline to classify and locate objects in an image. In the modified MobileNetv2 architecture, a squeeze and excitation (SE) block is added to the bottleneck residual block. This modified block is used as the basic block for the MobileNetv2 network. The SE block can exploit the channel-wise inter-dependencies in the output features maps of the depth separable convolution layer (in the bottleneck residual block). Experimental results on the VOC 2007 dataset confirm that with the addition of SE block, there is an increase in the mean average precision (mAP) of approximately 1%. The proposed pipeline achieves 56.5% *on the VOC 2007 test set for the category 'birds'.*

1. Introduction

Object detection is a fundamental and challenging problem in computer vision and is an active area of research. Object detection is a computer vision and image processing task for detecting objects of a particular class, such as human beings, buildings, cars, or animals, in digital images and videos. Object detection combines two tasks of image classification and object localization. Image classification involves predicting the class of an object in an image or video. In contrast, object localization refers to identifying the location of one or more objects in an image by drawing an abounding box around their extent. So, the object detection task locates objects with a bounding box and the class of the found object in the image. The input for this task is an image with one or more objects, such as a photograph. The output is one or more bounding boxes with coordinates of a point, width, and height and a class label for each bounding box. Object detection is applied in a wide range of applications such as robot vision, autonomous driving, security, consumer electronics, intelligent video surveillance, and augmented reality. As the field of application for object detection increases over time, it has become more important to develop algorithms that can detect objects in images and videos faster and more accurately.

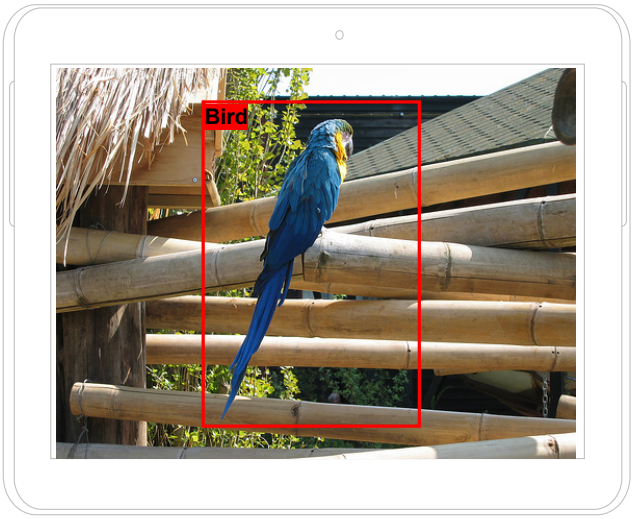

Figure 1. An overview of the Object Detection Algorithm realized on a mobile device.

Object detection, in recent times, has been improved significantly with the introduction of deep learning techniques for learning powerful feature representations automatically from the data [6, 10] and the GPUs for reducing the training time. Currently, object detectors in the object detection

algorithm belong to two different categories. One is a two-stage detector such as Faster R-CNN [16]. The second is a single-stage detector such as YOLO [13], Single Shot Detector (SSD) [12]. For Faster R-CNN, the first stage of the two-stage detector is a Region Proposal Network that proposes candidate object bounding boxes. The second stage is the Region of Interest pooling layer that extracts features by pooling operation from each candidate box for classification and bounding-box tasks.

On the contrary, single-stage detectors propose predicted boxes from input images directly without the region proposal step. Hence, single-stage detectors are time-efficient and are suitable for real-time devices [9]. In comparison, two-stage detectors have high localization and object detection accuracy, while single-stage detectors have high inference speeds. Different versions of YOLO (you only look once) algorithms proposed in [13, 14, 20, 19] improves the detection speed by combining the two-stage tasks of sorting and identifying candidate box as in Faster R-CNN. Still, the networks have a large number of parameters that make the detection speed slow. The network should be fast and accurate to realize the operation of object detection in real-time tasks on small devices. So, there is a need for a small deep learning model that can be hosted on small devices such as mobile devices, raspberry pi and can perform object detection on images and videos in real-time.

My contributions to this project can be summarized as follows. (**1**) Implement SSD architecture from scratch with modified MobileNet v2 network as its backbone for learning feature representations instead of VGG architecture as used by the original SSD paper [12]. (**2**) I propose a modified MobileNet v2 architecture network by adding a squeeze and excitation block between each inverted residual block. (**3**) Train and evaluate the proposed network pipeline from scratch on PASCAL Visual Object Classes Challenge (VOC) 2007 data set [1].

2. Related Work

In this section, I would discuss the previous works that have been done in the domain of deep learning and object detection.

Due to rapid development in autonomous vehicles, video surveillance, face detection, and anomaly detection, it has become more important to enhance the object detection algorithms for better performance in less time. Girshick et al. [3], in 2014, proposed a region-based convolutional neural network (R-CNN), which uses a region-based recognition method to detect objects. In 2015, two improved methods of R-CNN, Fast R-CNN [2], and Faster R-CNN [16] were proposed by Girshick et al. and Ren et al. that are focused on the end-to-end detection of targets. These models are based on two-stage algorithms that performed better than the traditional algorithm in terms of accuracy. However, the detection speed of these models are still too slow to be used in real-time applications.

Redmon et al., in 2016, introduced a one-stage algorithm, YOLO [13] and YOLO9000 [14]. In the YOLO pipeline, less than 100 bounding boxes per image are predicted, while the R-CNN pipeline uses a selective search that predicts 2000 region proposals per image. Further, YOLO frames detection task as a regression problem, so a unified deep learning architecture is used to extract features from input images directly to predict bounding boxes and class probabilities. Due to these techniques introduced in YOLO, the detection speed of the network increased several folds (45 frames per second) when compared to R-CNN (0.5 frames per second) and Faster R-CNN (7 frames per seconds). Also, an improved version of this algorithm was introduced in [15] by Redmon and Ali Farhadi. The latter greatly improved the effects of detection of small objects when compared to YOLO9000. YOLOv3 uses multi-label classification to adapt to datasets containing several overlapping labels.

Further, YOLOv3 proposed the utilization of three different scale feature maps to predict the bounding box effectively. Darknet-53 is proposed by YOLOv3, a deeper and robust feature extractor is inspired by ResNet architecture. Tiny-Yolo was introduced in 2017 and is widely used due to its high detection speed and less memory consumption than traditional YOLO algorithms. However, it is still challenging to use this algorithm for real-time applications on devices without GPUs. Zhao et al. [22] proposed a multi-level feature pyramid network to meet a large variety of scale variations across object instances by constructing more effective feature pyramids. Like feature pyramid networks [11], multi-level features extracted from multiple layers in the backbone are fused as the base feature. The base features are then fed into a block comprising of alternating joint Thinned U-shape Modules and Feature Fusion Modules, followed by the decoder layers of TUM as the features for the next step. Finally, a feature pyramid containing multi-level features is constructed by integrating the decoder layers of equivalent scale. Liu et al. proposed an SSD [12] algorithm to realize the regression detection of the whole image in a single pass (single shot) that improved the speed. However, the accuracy of the detector for detecting small objects is significantly decreased [23].

In 2017, MobileNet [7] by Howard et al., especially for mobile and embedded vision applications. In 2018, an improved MobileNetv2 [18] was proposed by Sandler et al., which introduced inverse residual networks to reduce the number of parameters significantly and hence increase the processing speed of the network. Several other efficient models were introduced during the same period, such as SqueezeNet [8] that proposed a Fire block containing a layer for squeezing the output and followed by an expan-

sion layer. ShuffleNet [21] was introduced by Zhang et al. that proposed group convolutions with shuffle operations.

In the next section, I would discuss and explain the basic blocks used in my pipeline, i.e., modified MobileNetv2 with Squeeze and Excitation block and SSD architecture.

3. Methodology

3.1. MobileNetv2 Network

MobileNet architecture [7] was introduced for making the deep learning models lightweight so that the models can then be realized on mobile and embedded vision applications. MobileNetv2 [18] network is an improvement on MobileNetv1 that introduced the use of depth separable convolutions with the incorporation of residual connection. In depth-separable convolutions, standard convolutions are factorized into two smaller operations: depth-wise convolution and point-wise convolutions to reduce the number parameters and mult-add operations in each convolution. In depth-wise convolutions, filters are applied to each of the input channels separately, i.e., for each input channel, a single channel convolution is used. In point-wise convolutions, a kernel of size 1x1 is applied to the output channels of the depth-wise convolutions after stacking them together.

Inverted residual block is introduced in the MobileNetv2 model. The inverted residual block comprises 3 different convolutional layers as follows:

1. The first layer is a bottleneck layer with a kernel size 1x1 with a rectified linear unit activation function, i.e., 'ReLU6'.

2. Second layer, in the inverted residual block, is a depth-wise convolution layer with a kernel size of 3x3 and with a 'ReLU6' activation.

3. The last layer is a linear (i.e., without any activation function) point-wise convolution layer with a kernel size of 1x1. It is claimed in [18] that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain.

In contrast to the original residual blocks proposed in [5], inverted residual blocks follow a narrow-wide-narrow approach. At first, the network has a bottleneck layer to narrow the input feature maps, followed by a depth-wise convolutional layer for expansion and a point-wise layer for making the output feature maps narrow again. The number of parameters at the depth-wise convolution layer is reduced significantly.

As a contribution to this project, as shown in figure 2, I introduce the fourth block of squeeze and excitation layer after the depth-wise convolution layer in the inverted residual block of MobileNetv2. The motivation behind introducing the squeeze and excitation block after the depth-wise

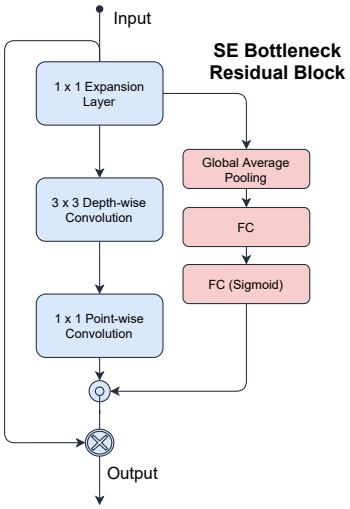

Figure 2. Shows the proposed SE Bottle Residual Block as a basic block for the SE MobileNetv2 architecture. ⊗ represents concatenation, and ⊙ represents element-wise multiplication.

convolution is to exploit channel-wise inter-dependencies in the output feature maps of the depth-wise convolution layer. The goal of squeeze and excitation networks is to improve the representational power of a network by explicitly modeling the inter-dependencies between different channels of the feature maps.

The SE block performs feature re-calibration through which it can learn to exploit the global information to identify and select important informative features and suppress less useful ones. For performing feature re-calibration, the input features are first passed through a squeeze operation. The squeeze operation averages the feature maps across a spatial dimension of the maps and produces a channel descriptor. The channel descriptor represents the embedding of the global distribution of the channel-wise features. This channel descriptor is achieved by using a global average pooling layer to generate channel-wise statistics. An excitation operation follows the squeeze operation. This operation aims to capture the channel-wise dependencies by learning weights respective to each channel based on its importance. In this operation, a fully connected layer followed by a ReLU activation is added to introduce necessary non-linearity in the network. The complexity of this layer can be adjusted by a reduction ratio 'r'. Then, a second fully connected layer is added with a sigmoid function to activates the channel-specific information. At last, learned activations for each channel is multiplied with the respective feature map of the convolutional block to re-weight the output of the SE block based on the importance of the features. This output is further fed directly to subsequent layers.

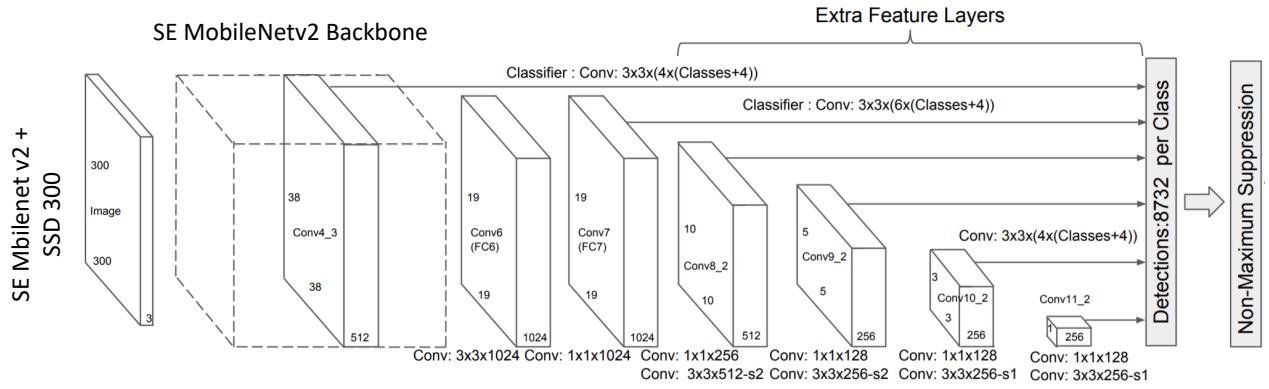

Figure 3. Shows the SSD300 architecture with SE MobileNetv2 as its backbone.

Input	Operator	Output
$h \times w \times k$	1×1 conv2d, ReLU6	$h \times w \times (tk)$
$h \times w \times (tk)$	3×3 dwise s = s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times (tk)$	linear 1×1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times (k^{'})$
$\frac{h}{s} \times \frac{w}{s} \times (k^{'})$	se block, r = 16	$\frac{h}{s} \times \frac{w}{s} \times (k^{'})$

Table 1. Shows the Bottleneck block transforming the input with k channels to $k^{'}$, where s stands for stride, t stands for expansion factor, and r stands for reduction ratio.

Input	Operator	t	c	n	s
$300^2 \times 3$	conv2d	–	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	bottleneck	6	1280	1	1

Table 2. Shows the Bottleneck block of MobileNetv2 network used as a backbone for SSD, where each line describes a sequence of 1 or more identical layers.

3.2. Single Shot Detector: SSD

Object detection aims to identify what objects are there in an image (i.e., classification) and the location of those identified objects in the image (i.e., localization). Single-shot detector introduced in [12] is a multi-scale sliding window detector that uses the deep CNN as its backbone (or basic network) to extract features that further perform both of the mentioned tasks. In sliding window detection, a local window slides across the input image and then identifies whether the sliding window contains an object of interest at each location. A multi-scale sliding window means that different sizes of windows are considered to increase the robustness of the detection. As stated earlier, deep CNN networks predict the precise location of the object in an image in addition to the object's class. Instead of just mapping the pixels to a vector of class scores, SSD maps the pixels of the image to a vector of four floating numbers that represents the bounding box. The scales and aspect ratio of the default boxes is given as follows:

$$S_k = S_{min} + \frac{S_{max} - S_{min}}{m - 1}(k - 1), k \in [1, m] \quad (1)$$

So let, there are m feature maps for predictions, then S_k is calculated for the k^{th} feature map. the scale of the lowest layer is (S_{min}) is 0.2 and the scale of the highest layer (S_{max}) is 0.9. All layers that are in between are spaced evenly.

The loss function used for training the SSD network consists of two terms: L_{conf} and L_{loc}. L_{conf} is the confidence loss which represents the softmax loss over multiple classes confidences (c), while L_{loc} is the localization loss that represents the smooth L1 loss between the predicted box (l) and the ground-truth box (g) parameters. The total loss functions is calculated as follows:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c)) + \alpha L_{loc}(x, l, g) \quad (2)$$

where N is the number of matched default boxes. x represents if the matched boxes belong to a category p, and the value can be $0, 1$. g represents the ground truth box; while c refers to the confidence of the checked object belonging to category p.

3.3. Problem Statement

As already discussed in the above sections, since SSD uses the single-stage detection technique and does not use the delegated region proposal network, they are time-efficient compared to R-CNN algorithms. However, as the SSD network uses the VGG pipeline to learn representations, it still has too many parameters that make it difficult to realize on mobile devices. As the MobileNetv2 networks are very time-efficient and explicitly designed for mobile devices, it would be good to use a combination of MobileNetv2 architecture with the SSD network. So, to overcome the above-mentioned issue, I have replaced the VGG backbone of the SSD network with the MobileNetv2 network. Also, to improve the MobileNetv2 architecture, I add a squeeze and excitation network in each inverted residual block. In the next section, I would discuss the details of the MobileNetv2 architecture along with the SSD pipeline.

3.4. Network Architecture

In this section, I would first discuss the details of the MobileNetv2 architecture and how the squeeze and excitation block is inserted in each of the residual blocks. Followed by this, I will discuss how the modified MobileNetv2 architecture is combined with the SSD layers.

As discussed in section 3.1, the basic building block of the MobileNetv2 architecture is a bottleneck depth-wise separable convolutions with residual connections. The first layer of the bottleneck residual block is a bottleneck convolutional layer with a 1x1 kernel size and ReLU6 activation function. This is followed by a depth-wise separable convolutional layer with a kernel size of 3x3 and ReLU6 activation function and a point-wise convolutional layer of kernel size 1x1. In the modified residual block, an SE block with a reduction ratio 'r' of 16. The width multiplier used for the expansion layer (i.e., depth-wise separable layer) is 0.75. The basic implementation structure of the modified residual block is illustrated in Table 1.

The overall architecture for modified MobileNetv2 contains a first convolution layer with 32 filters. Nineteen modified residual bottleneck layers follow this. Each row of Table 2 mentions a sequence of one or more identical layers that are repeated n times. All the layers at a particular row of the table have the same number of output channels c. The first layer at each row has a stride of s, and all others use stride 1. All spatial convolutions at each stage use a kernel size of 3×3. As described above, the expansion factor of t is always applied to the input size. For combining it with the SSD layers, the first layer of SSD is attached to the expansion of layer 15 (output feature map with the stride of 16). The second and the rest of the SSD layers attached after the last layer with an output stride of 32. The details of the complete pipeline are mentioned in the figure 3.

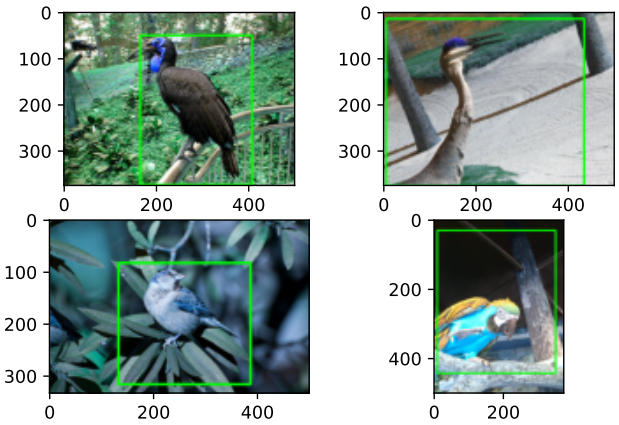

Figure 4. Detection samples from the VOC 2007 test dataset with SE MobileNetv2 SSD300 model

4. Implementation

4.1. Dataset: PASCAL VOC 2007 & VOC 2012

The PASCAL VOC datasets [1] were developed from the year 2005 to 2012 with a motive to create and maintain a series of benchmark datasets for object detection and segmentation tasks. The datasets (VOC 2007, VOC 2012) consist of 20 object categories and around 11,000 images. Some of the object categories in the PASCAL VOC datasets are person, birds, bicycle, bottle, dog, etc. The 20 categories are considered to be a subset of 4 main categories: vehicles, animals, household objects, and people. In the dataset, over 27,000 object instance bounding boxes are labeled. Out of these 27,000 labeled boxes, 7,000 have detailed segmentation. VOC 2007 and VOC 2012 are taken as training datasets in this project, while VOC2007 test sets are used for testing.

4.2. VOC 2007 Metric

Usually, there are three primary criteria for evaluating detection algorithms: detection speed in Frames per second (FPS), precision, recall. A commonly used metric is Average Precision that is derived from precision and recall. AP is usually evaluated for a specific category, and the mean AP (mAP) is the average overall object category. In the VOC 2007 evaluation criteria, the interpolated average precision is used to evaluate classification and detection. This criterion is designed to penalize the algorithm for missing object instances, duplicate detections, and false-positive detections [9]. The mean average precision (mAP) is used to evaluate the performance of the target detection model for only one category, 'birds' in the dataset. The precision and recall

Method	Dataset	Training Iteration	mAP (Category: Birds)
SSD300 [12]	VOC 2007 & VOC 2012	200k	72.3
MobileNetv2 + SSD300 [17]	VOC 2007 & VOC 2012	200k	77.43
MobileNetv2 + SSD300 [Ours]	VOC 2007 & VOC 2012	40k	55.7
SE MobileNetv2 + SSD300 [Ours]	VOC 2007 & VOC 2012	40k	56.5

Table 3. Compares the performance of our implementation with the state-of-art implementations of SSD300.

scores are calculated as follows:

$$Recall(t) = \frac{\sum_{ij} 1[s_{ij} \geq t] z_{ij}}{N} \tag{3}$$

$$Precision(t) = \frac{\sum_{ij} 1[s_{ij} \geq t] z_{ij}}{\sum_{ij} 1[s_{ij}} \tag{4}$$

where, t represents the threshold to judge the intersection over union (IoU) between predicted box and ground truth box, i represents the idenx of i-th image and j represents the index of j-th object in the image. For VOC metric, the value of t is set to 0.5.

4.3. Implementation Details

The proposed model is implemented using the PyTorch framework on a computer with an NVIDIA GeForce RTX 2070 Super GPU. As mentioned above, for training the modified mobilenetv2 architecture with SSD pipeline, training data from VOC 2007 and VOC 2012 is combined with learning the feature maps while the trained model is tested on the test set from VOC 2007. 'Xavier' method is used to initialize all the parameters of convolutional layers [4]. The developed network architecture was trained for 40k iterations. A learning rate of 10^{-3} was used with the cosine annealing technique in which the learning rate is relatively decreased to a minimum value before being increased rapidly again. A warm-up learning rate was used for the first 2000 iterations.

4.4. Results

This section will discuss the results, listed in Table 3 of my implementation of the SSD300 network with mobilenetv2 and SE mobilenetv2 as its backbone. Along with it, I will compare my results with the state-of-the-art algorithms such as the traditional SSD300 network architecture [12] and PyTorch implementation of mobilenetv2 + SSD300 in [17]. Figure 4 shows the detected bounding boxes for some test images from bird categories. It can be observed that the model is able to identify the birds in the image most of the time; however, the bounding box around the birds could still be improved. It is interesting to note that by the addition of SE block at each bottleneck residual block of mobilenetv2 (row 4), there is a performance increase of approximately 1% when compared to our own implementation without the SE block (row 3). The SE mobilenetv2 + SSD300 achieved a mean average precision of 56.5%. This demonstrates that adding an SE layer between each bottleneck residual block helps in identifying channel-wise inter-dependencies. It can be observed from the table that the mobilenetv2 + SSD300 by [17] still outperforms the traditional SSD300 architecture with VGG16 as its backbone (basic block) and our method by a margin. It can be argued that a better performance for SE mobilenetv2 + SSD can be achieved if they are trained to full convergence, similar to the state-of-the-art methods. However, due to resource constraints, it was impossible to train the model from scratch to full convergence.

5. Discussion and Conclusion

In this report, I proposed a modified version of mobilenetv2 architecture used to learn feature maps for the main SSD300 pipeline for object detection. A squeeze and excitation block is added to each bottleneck residual block of mobilenetv2 architecture. The motivation behind adding the SE block was to exploit channel-wise inter-dependencies in the output feature maps of the depth separable convolution layer. Based on the analysis, it can be understood that adding the SE layer to the network helps the mobilenetv2 architecture to learn better feature maps for the SSD300 pipeline. It is interesting to note that my implementation of the SE mobilenetv2 + SSD300 pipeline achieved an average mean precision of 56.5% outperforming the pipeline without the SE block. However, the performance of the proposed model could not reach much closer to the state-of-the-art implementations. This could be because the model was not trained until full convergence. It would be interesting to observe how the data augmentation changes the performance of the proposed pipeline. Also, as the motive was to create a deep learning-based object detection algorithm that can be realized on small devices, it is interesting to note that even with the addition of SE block at each bottleneck residual block, the overall size of the model is just below 12.2MB which is just 1MB more than the pipeline without the SE block.

References

[1] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2, 5

[2] Ross Girshick. Fast r-cnn object detection with caffe. In *Proceeding of the 13th European Conference on Computer Vision*, 2014. 2

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[4] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 6

[5] 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. 3

[6] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 1

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3

[8] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 2

[9] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019. 2, 5

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 1

[11] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2

[12] 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. 2, 4, 6

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[14] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 2

[15] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. 2

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 2

[17] dong Robin. Ssd pytorch. https://github.com/RobinDong/ssd.pytorch, 2019. 6

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2, 3

[19] Sheng Wang. Substation personnel safety detection network based on yolov4. In *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pages 877–881. IEEE, 2021. 2

[20] Pengyi Zhang, Yunxin Zhong, and Xiaoqiong Li. Slimyolov3: Narrower, faster and better for real-time uav applications. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[21] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 3

[22] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9259–9266, 2019. 2

[23] Zuopeng Zhao, Zhongxin Zhang, Xinzheng Xu, Yi Xu, Hualin Yan, and Lan Zhang. A lightweight object detection network for real-time detection of driver handheld call on embedded devices. *Computational Intelligence and Neuroscience*, 2020, 2020. 2