Face Mask Detection

XIANG YAO NG
2019280437
Electronic Engineering, Tsinghua University
huang-xy19@mails.tsinghua.edu.cn

WOOMIN MYUNG
2019280362
Electronic Engineering, Tsinghua University
mingym19@mails.tsinghua.edu.cn

Abstract

In the time of a COVID-19 pandemic, people are more concerned of their health and are more vigilant in taking care of their own safety and wellbeing. In order to combat the spread of the virus, it was crucial for people to practice social distancing and constantly wear masks. Hence, it was practical to develop algorithms that identify people that do not wear mask as well as wearing a mask. In this paper, we will explore three approaches: YOLO, YOLOv2 and SSD.

Keywords: Face Mask, Object Detection, YOLO, YOLOv2, SSD

1. Introduction

COVID-19 is a very potent respiratory disease that has negatively impacted the daily lives of many people. The virus has spread all over the world with a high infection rate and considerably low fatality rate. However, it remains extremely lethal to older people and can cause serious respiratory problems even after the patient has recovered. Hence, it is necessary to wear face mask when travelling outdoors and maintain social distancing to mitigate the spread of the virus. This change of daily habit cause another repercussion, which is the surveillance camera is not able to properly detect faces with mask since it was not programmed to do so. Therefore, there is a need to develop algorithms to tackle the face mask detection problem.

To detect face mask, there are two categories of ways to approach this. First, the model consist of a region proposal network that performs object localization and a classifier for detecting objects in the proposed regions. Second, the model computes both localization and detection in a network.

We can view this problem as a category of object detection problem, since many algorithms has been developed to solve these problems. Machine learning has been explored for many years, and it was not until a Deep Learning (DL) algorithm won an Imagenet competition by a large margin that DL algorithms are now the mainstream method to solve computer-vision related problems, including object detection. Recent popular object detection algorithms include Faster RCNN, SSD, YOLO, RetinaNet etc.

In this paper, we will explore the SSD, YOLO and YOLOv2 to solve the face mask detection problem.

2. Dataset

Dataset Source The dataset is taken from the AIZOO website [1]. There are 7959 images consist of faces with masks worn and no masks. In the dataset, 6120 and 1839 images are designated as training images and validation images from the ”train” folder and ”val” folder respectively.

Preprocessing To clean the dataset, we set rules to filter the data:

1. If the bounding box or label is missing, the image data is removed.
2. If the image size is either missing or is set to 0, the image data is also removed.

Dataset splits We split the data in the following format: first 6000 images are used as our train dataset, the next 240 images are taken as validation set and the remaining 1719 images are assigned as test set. The split format can be illustrated in the following Figure 1.

Note: The testing of hyperparameters are done on validation set.

The distribution of the images classes are recorded, which can be observed from Figure 2.

3. Model Architectures

3.1. Faster RCNN

NOTE: WHILE WE MANAGED TO BUILD THIS MODEL AND IS ABLE TO TRAIN THE MODEL UNTIL THE END, THE RESULTS ARE NOT SATISFYING. HENCE, AFTER TAKING TWO WEEKS TO WORK ON THIS, WE WORK ON OTHER MODELS INSTEAD.

Faster RCNN is built on top of Fast RCNN, a form of region-based convolutional neural network (RCNN). While Fast RCNN are very adept in detecting objects with highly accurate results, time spent on region proposals are too long via the Selective Search method to be feasible for real-time detection with a 2 seconds per image processing speed on CPU implementation. In order to overcome this bottleneck, Region Proposal Networks (RPNs) are introduced that share convolutional layers with the Fast

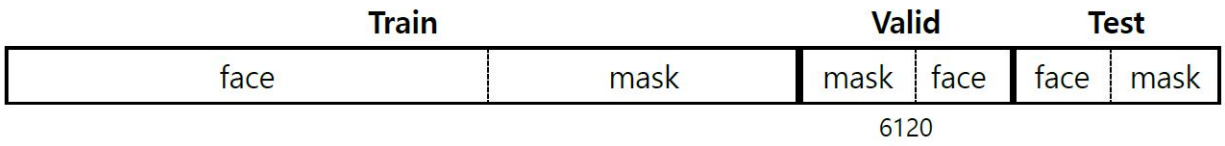
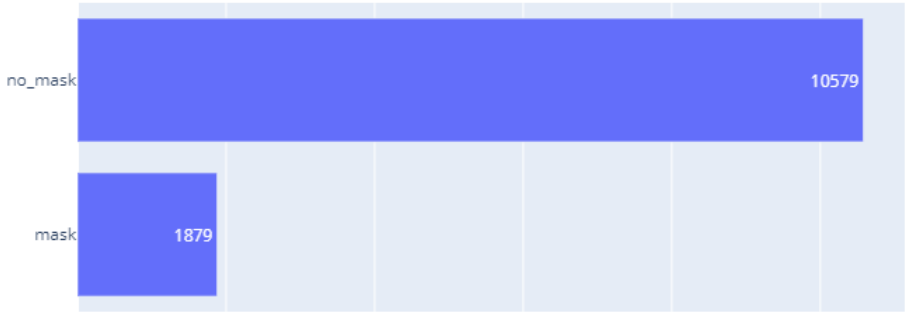
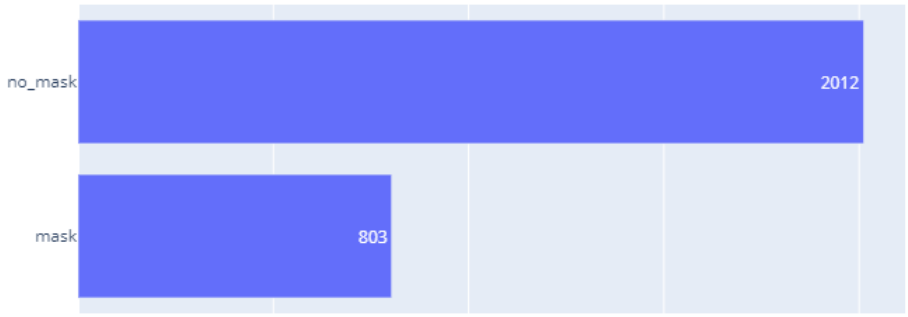


Figure 1: Dataset Split



(a) Counts of Classes (Train folder)



(b) Counts of Classes (Val folder)

Figure 2: Dataset Distribution

RCNN object detection networks. An illustration of the model architecture is shown in Figure 3.

RPN RPN takes an image and outputs a set of object proposals. The object proposals will then be given to a Fast-RCNN network. The RPN uses a sliding window to move across all feature maps.

Anchors For each sliding-window location, anchors are used, which as a box of pre-defined scale and shape. In the default setting from the paper, 9 different anchors are used with 3 different scales and 3 different ratios. Each anchor is used to predict both regression task for predicting bounding boxes and classification task for predicting object labels and produced 4 scores for regression (xmin, ymin, xmax, ymax) and 2 scores for classification (object or not object).

Alternate training In the paper, the RPN is first trained, and then the proposals is used to train the Fast RCNN. The trained Fast RCNN is then used to initialize RPN, which the training repeats alternatively.

4-Step Alternating training In the alternating training algorithm, there are 4 steps. First, the RPN is trained, initialized by a pre-trained ImageNet model. Second, the Fast RCNN is trained using the proposalss by the RPN. Third, the trained Fast RCNN is used to initialize a new RPN where only layers unique to RPN layers are fine-tuned and the shared convolutional layers fixed. Forth, the RPN is is used to train the unique layers in Fast RCNN, while fixing the shared convolutional layers.

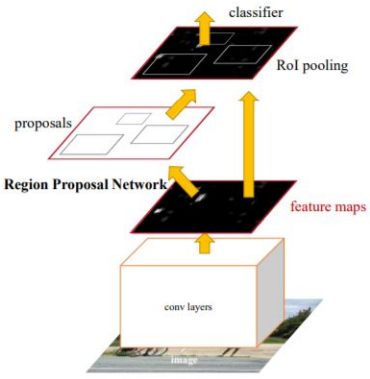


Figure 3: Model Architecture of SSD [3]

3.2. YOLO

In the purpose of emulating human visual system to glance at an image and instantly know what objects in their sight, the YOLO model strives to predict object detection in an with one go. A single convolutional network simultaneously predicts object's bounding boxes and its respective classes. The model architecture are illustrated in Figure 4.

Grid The input image is divided into $S \times S$ grids. If center of object fall into these grids, the grid cell is responsible for detecting the object. Each grid should predict bounding boxes and its corresponding confidence scores.

Network Design The network has 24 convolutional networks followed by 2 fully connected layers. 1×1 reduction layers are followed by 3×3 convolutional layers.

Training The convolutional layers are pretrained on the ImageNet. Linear activation function is used for the final layer while other layers uses leaky ReLU activation function. The output is optimized by sum-squared error with a higher loss on bounding box prediction and lower loss on confidence predictions of objects that does not contain objects.

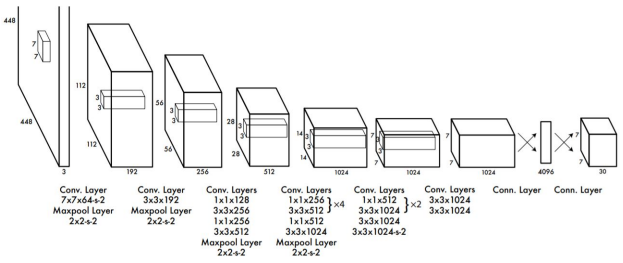


Figure 4: Model Architecture of YOLO [4]

3.3. YOLOv2

The model YOLOv2 was referred from the paper "YOLO9000: Better, Faster, Stronger" [5]. This paper introduces both the YOLOv2 and YOLO9000 architectures. From the title, it was easy to observe that the model was built on top of the YOLO foundation with improved algorithms, in particular, the "9000" in the naming means the model is enabled to detect up to 9000 different classes of objects. YOLOv2 and YOLO9000 has the same model architecture. As the objective of our task is Face Mask detection, which 3 classes of detection are considered (Face mask, Face and Background). Hence, YOLOv2 is picked to approach this task.

Improvements from YOLOv2 There are 8 main improvements to YOLOv2 from YOLO:

1. Batch Normalization

   In the original YOLO, Dropout is used to mitigate overfitting. In YOLOv2, Batch Normalization is used instead. This allows better loss convergence.

2. High Resolution Classifier

   Most state-of-the-art detection algorithm's model backbone uses ImageNet' pretrained model, where the input of the classification model is 224×224, while YOLO's input is 448×448.

3. Convolutional with Anchor Boxes

   The final 7×7 fully connected layer of the backbone of YOLO contains the Anchor data. Each grid contains two Anchor box. However, it will be difficult to consider the accuracies of various sizes and shapes of the bounding box in this form. Hence, in reference to the Faster RCNN's hand-picked priors method, convolutional layers replaces full connected layers instead. On the other hand, the feature maps of YOLOv2's input image' sizes are fixed to odd number in order to make sure the image has a center point.

4. Dimension clusters

   We talked about using hand-picked priors in the previous item. However, if the model follows Faster RCNN's method to pick priors, no matter the distribution of ground truths, as long as the input image's size are the same, all its priors are also the same. Hence, YOLOv2 implements the k-means clustering method to choose size and shape of priors from ground truth of the train set. The distance metrics used is the IoU, which can be described as follows:

   $$d(box, centroid) = 1 - IoU(box, centroid)$$

5. Direct Location Prediction

   The prediction bounding boxes are calculated as shown if Figure 6.

   where Sigmoid is used to contain the center of the bounding box in the grid.

6. Fine-Grained Features

   The classic problem of YOLO is the fact that the accuracy of detection of objects that are near to the camera or small objects are slightly worse. This is because the model predicts detections on 13×13 feature map only instaed of various sizes of feature maps like in Faster R-CNN and SSD. In order to tackle this issue, however, YOLOv2 introduces a passthrough layer that concatenates the higher resolution

features to low resolution features by stacking into different channels. This turns the $26 \times 26 \times 512$ feature map into a $13 \times 13 \times 2048$ feature map.

7. Multi-Scale Training

   As mentioned earlier, YOLOv2 replaces fully connected layers with convolutional ayers. Hence, there is no restriction on the size of input images. This allows training of multi-scale images, which improves the robustness of the model.

   Note: Through experiment by following the original paper, if the size of images changes every 10 batches, the loss convergence become slower. Since the improvements is not significant, the practice is ignored.

8. DarkNet-19

   The backbone of YOLO is GoogleNet. However, YOLOv2 uses a novel DarkNet-19. Its structure is illustrated in Figure 5. Its structure is simpler, which reduces computation resources while maintaining good performance.

Loss Function The equation of the loss function of YOLOv2 is shown in 7. YOLOv2 uses priors, hence, IoU will need to be computed with the ground truth, and only the box with the largest confidence score is included in calculation of object loss, which the other $k-1$ priors will be included in the no obkect loss calculation. However, this method will cause imbalance negative samples problem. Therefore, in the experiment, the model excluded the samples with IoU bigger than threshold (such as 0.6) for no object loss calculation.

Note: In the original paper, the exact details of the loss function is not specified, the format of the loss function is inferred from the released code of Github [2]. The experiment also refers to the some functions in https://github.com/yjh0410/yolov2-yolov3_PyTorch.

Technique Detail

Training phase The hyperparameters used for the experiment is as follows:

1. First 30 epoch: $learning\_rate = 5 \times 10^{-4}$, $\lambda_{coord} = 1, \lambda_{obj} = 5, \lambda noobj = 1, \lambda_{class} = 1$
2. Next 80 epoch: $learning\_rate = 5 \times 10^{-5}$, $\lambda_{coord} = 10, \lambda_{obj} = 10, \lambda noobj = 0.5, \lambda_{class} = 0.5$
3. Next 120 epoch: $learning\_rate = 1 \times 10^{-5}$, $\lambda_{coord} = 1, \lambda_{obj} = 1, \lambda noobj = 5, \lambda_{class} = 5$
4. Next 150 epoch: $learning\_rate = 1 \times 10^{-5}$, $\lambda_{coord} = 20, \lambda_{obj} = 5, \lambda noobj = 3, \lambda_{class} = 3$ $momentum = 0.9, weight\ decay = 5 \times 10^{-4}$

Testing phase

1. Pretrain the YOLOv2 with DarkNet-19.
2. Use direct location prediction to get predicted bounding boxes.
3. Ignore bounding boxes with low confidence score (such as 0.1)
4. Perform Non-Maximum Suppression (NMS) on the rest of the bounding boxes to get the final result.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 5: DarkNet-19 [5]

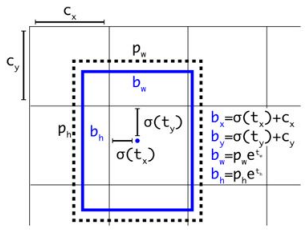


Figure 6: Box Prediction [5]

$$\lambda_{\text{coord}} \sum_{i=0}^{w*h} \sum_{j=0}^{n} 1_{ij}^{\text{obj}} (2 - w_i * h_i)[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (w_i - \hat{w}_i)^2]$$
$$+ \lambda_{\text{obj}} \sum_{i=0}^{w*h} \sum_{j=0}^{n} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{w*h} \sum_{j=0}^{n} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$
$$+ \lambda_{\text{class}} \sum_{i=0}^{w*h} \sum_{j=0}^{n} 1_{ij}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Figure 7: Loss Function

GPU	Operating System	Coding Environment
TiTan X	Ubuntu 16.04	refer to env.yml

Table 1: Environment

3.4. Single-Shot Multibox Detection (SSD)

An simple illustration of the model architecture is described in Figure 8.

Base network The model uses a truncated version VGG-16 network as the base network. It is then expanded by adding auxiliary structures that help to produce detections with the following key features.

Multi-scale feature maps Convolutional layers are added at the end of the base network. The size of layers then decreased in stages, which allows prediction of detections at different scales.

Convolutional Predictors For each added convolutional layers, predictions of detections can be produced, because the laters can represent the original images at different scales. Using a fixed-size filter operating on different feature maps, objects of various sizes can be detected.

Default boxes and aspect ratios By using pre-defined boxes at specific positions, aspect ratios and scales, offsets of ground truths relative to the default box, and the pre-class scores for each of the boxes can be predicted, which the loss can then be computed between these boxes and ground truths.

Hard Negative Mining Since the majority of predictions do not actually contain objects, a method called Hard Negative Mining is introduced. The idea is to use the predictions that are the most wrong about the image to train the model. This allows the model to learn more rapidly so that it would reduce the likelihood of producing more negatives at next iterations.

Non-Maximum Suppression Since it is unfortunate that multiple priors may overlap, it is crucial to filter out the overlaps that has the least probability. By using Non-Maximum Suppression, duplicates of the same object can be removed and the prediction with the maximum score is retained.

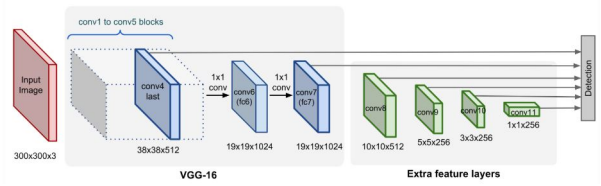


Figure 8: Model Architecture of SSD

	mAPs	AP (No Mask)	AP (Mask)
Normal	0.853	0.819	0.887
$[2, 1/2]$ only	0.847	0.812	0.883
$[3, 1/3]$ only	0.853	0.823	0.884

Table 2: Hyperparameter of SSD

4. Results

For Faster RCNN, no results will be released as we have only made attempt to build the model.

5. Model Analysis

5.1. Model Comparison

To compare the model of each approach, we have listed a table of comparisons in Figure 3.

We have also plotted Precision-Recall (PR) curves for the all the approaches in Figures [12, 13] (SSD), [14, 15] (YOLO) and [16, 17] (YOLOv2).

5.2. Hyperparameters (SSD)

We consider the anchor's aspect ratios as hyperparameters of the SSD. In our experiment, we keep only either $[2, 1/2]$ or $[3, 1/3]$ aspect ratios and compare it with the normal model to observe its performance. Only threshold 0.5 is considered, which is shown in Table 2.

From the table, we can see that keeping only $[3, 1/3]$ does not affect the performance too much, while losing that aspect ratio slightly decrease the performance. We propose that it is the case because most of the objects in the images are comparably large objects, hence bigger aspect ratio is needed to effectively detect the objects.

6. Evaluation Analysis

6.1. Prior Selection (YOLOv2)

As mentioned from the paper, unlike hand-picked priors by Faster RCNN, k-means clustering is trained on the training set to automatically find good priors. A figure shows the k-means clusters in 9.

From the result, we conclude that the suitable anchor coordinates are as follows 10.

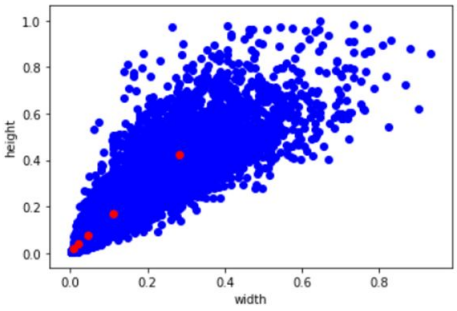


Figure 9: k-means clustering for priors

```
anchor = [[0.3173, 0.4639],
          [0.1370, 0.2163],
          [0.0673, 0.1082],
          [0.0312, 0.0553],
          [0.0144, 0.0264]]
```

Figure 10: Anchor centroids

	TP	FP	n_objects
no_mask	1632	3458	1811
mask	738	551	790

Table 4: TP and FP analysis of SSD

6.2. True Positives and False Positives Analysis (SSD)

We plotted the cumulative sums of True Positives (TP) and False Positives (FP) of the test set, as shown in Figure **??**.

	Threshold	mAPs	AP (No Mask)	AP (Mask)
YOLO	0.5	0.404	0.334	0.472
	0.7	0.133	0.083	0.183
	0.9	0.004	0.003	0.004
	[.5:.95]	0.174	0.123	0.228
YOLOv2	0.5	0.347	0.290	0.403
	0.7	0.145	0.113	0.176
	0.9	0.012	0.0012	0.023
	[.5:.95]	0.163	0.122	0.206
SSD	0.5	0.853	0.819	0.887
	0.7	0.603	0.567	0.639
	0.9	0.017	0.019	0.014
	[.5:.95]	0.527	0.498	0.556

Table 3: Evaluation Results of each approach

We can observed that there are many FP for no_mask detected while fairly low FP for mask detected. We can also observed that TP for both no_mask and mask are quite high when the number of objects is considered. From here, we can tell that there are maybe many duplicate detections for no_mask. We can see that this is indeed the case when we discussed the next section.

6.3. Case Analysis (SSD)

We will now make a case by case analysis on the detection results of SSD to dissect and explain the potentials and limits of the approach. The cases will be referred to the Figure 11.

In (a), we can see multiple detections has been made on the same face. This is caused by occlusion by the shelf stand, which blocks part of the face. As a result, the model was not able to circumvent the occlusion to picture a full face from the image. However, since the divider (shelf stand) splits the face into two parts, the model was somehow able to retrieve certain features of a face such as ears, nose, eyes, mouth etc from both sides to determine that it they are faces.

In (b), multiple detections are made on the man's face while no detection is made on the lady's face. I conjecture that this is because when a face does not face the camera, it cannot properly read all the features require to be a face, which causes it to make multiple detections based on separate features or no detection at all.

In (c), a hand blocks a face in the image. Although the model manages to detect the face as a no_mask, it also identify the hand as a face. It might be that the model mistakenly "thought" that the hand is a feature of a face such as a face cheek due to similar skin colour. Coupled with other face features like eye and nose, it combined the hand, eye and nose as a face.

In (d), a clear face image was present, however, it was not detected. We think that it was due to the grey scale image. The lack of distinction of pixel colours led to the model not being able to distinguish between the environment and a face.

In (e), this is a clear mask detection, but it also included another no_mask detection. This result was present in some other images as well. Generally, it happened with images where white mask is worn with a white skin colour.

In (f), a no_mask was detected on a hand of a man, while a detection is made on two faces combined. The detection of the hand is again a case of mistakenly identify a hand to a feature of a face due to similar skin colour. The detection on multiple faces however is maybe caused by the model combining features of different faces to make up a face. Hence, it can be seen clearly that the model detects features of a face rather than trying to detect a whole face.

7. Code Reference

For SSD, most of the code are heavily referred from https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection, especially in model.py, utils.py, train.py, detect.py, and eval.py. There are various level of reference. In the code section, there are comments to explain this, written on top of each functions and class. The term "taken" means that the code is copied without any modification from its source code, the term "slightly modified" means that most of the original code is retained as new items are added to suit the problem's needs, and the term "heavily modified" means that most of the code have been modified.

8. Conclusion

We have explored some of the approaches of object detection algorithm that can help to detect mask and no_mask with high accuracy. However, we did not managed to work out our Faster RCNN in the end. In the future, we would like to improve our models as well as test out different models just to see how it approach this detection problem and how the performance would different from our current models.

References

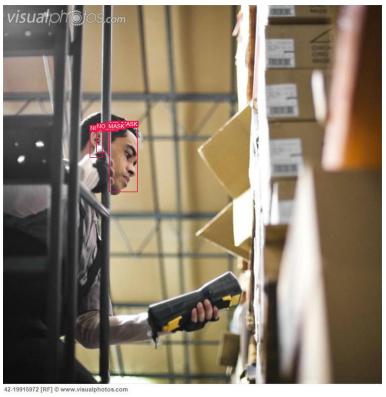
[1] FaceMaskDetection. https://github.com/AIZOOTech/FaceMaskDetection. Accessed: 2020-06-05. 1

[2] YOLOv2 and YOLOv3 Pytorch. https://github.com/yjh0410/yolov2-yolov3_PyTorch. Accessed: 2020-06-14. 4

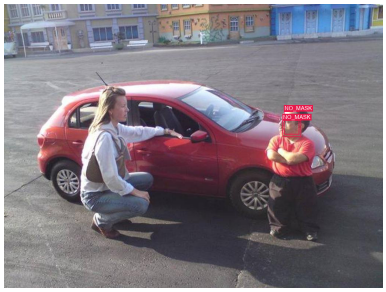
[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 3

[4] J. Redmon, S. Divvala, R. Girshick, and A. 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. 3

[5] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 3, 4



(a)



(b)

(c)

(d)

(e)

(f)

Figure 11: Labeled images (SSD)

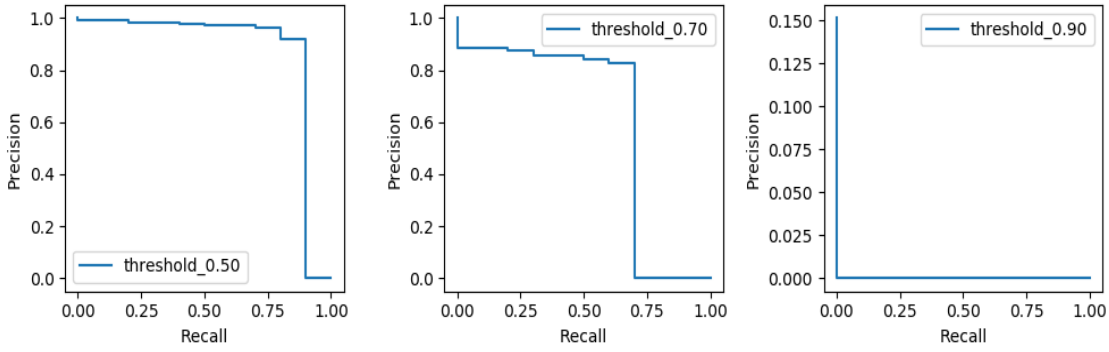


Figure 12: PR curve of Face Mask (SSD)

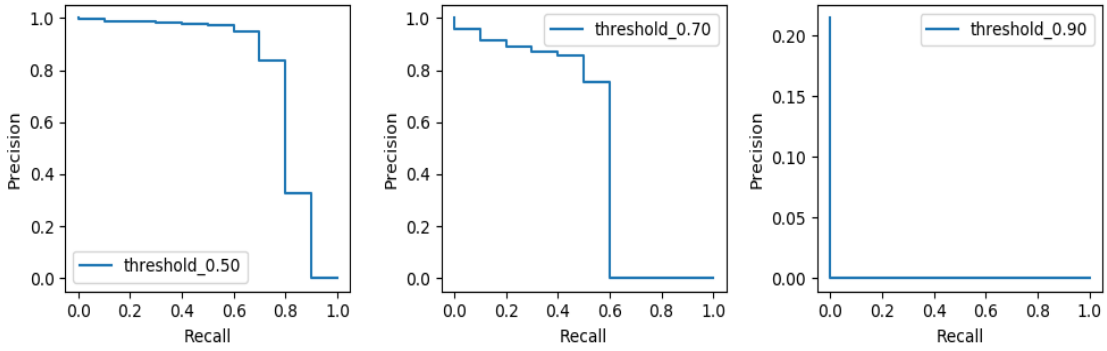


Figure 13: PR curve of Face (SSD)

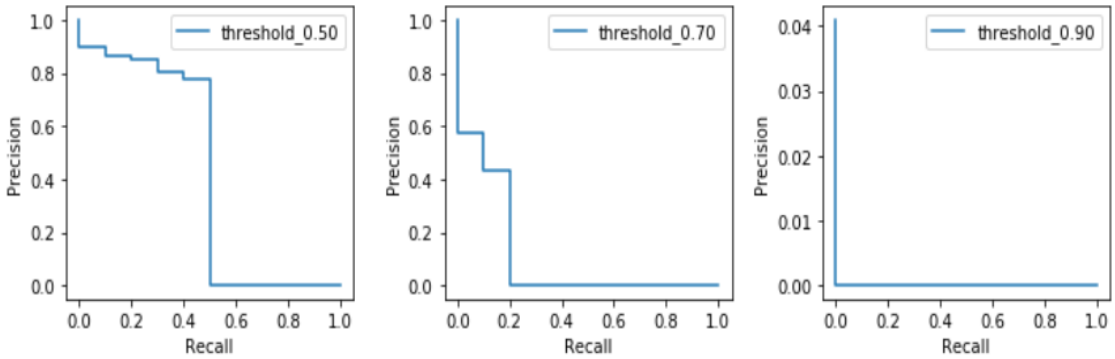


Figure 14: PR curve of Face Mask (YOLO)

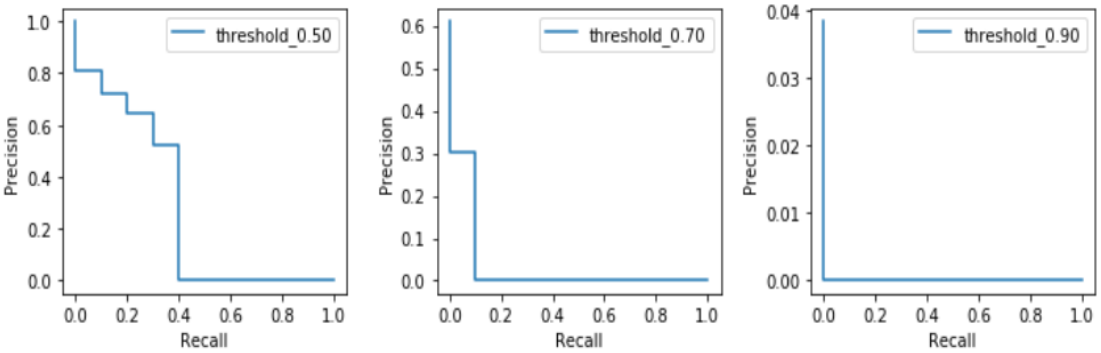


Figure 15: PR curve of Face (YOLO)

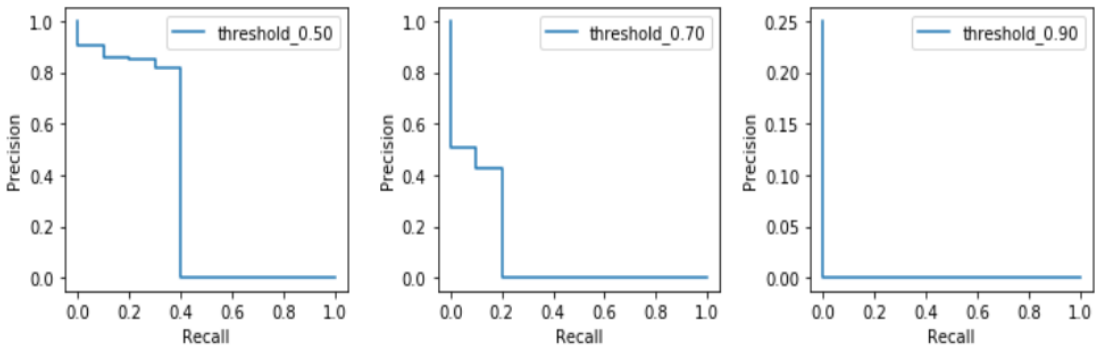


Figure 16: PR curve of Face Mask (YOLOv2)

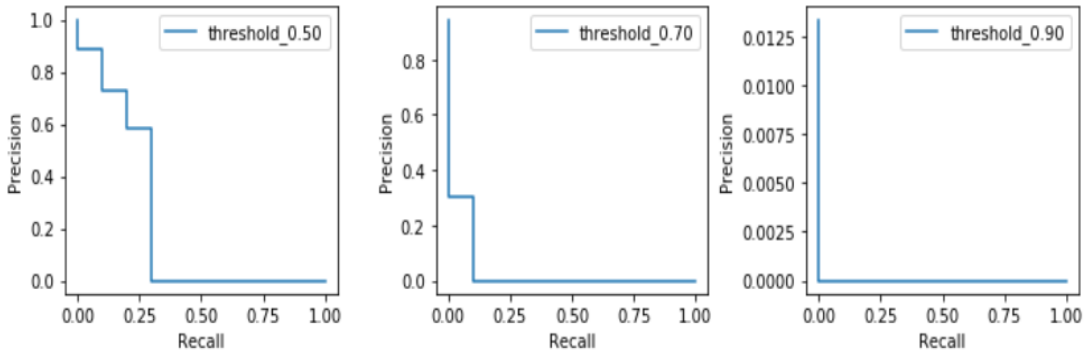


Figure 17: PR curve of Face (YOLOv2)