Face mask detection using Faster R-CNN and YOLOv5

Pham Thanh Hung
20194437
hung.pt194437@sis.hust.edu.vn

Abstract

Billions of people nowadays are affected by Covid19. The uncontrollable condition may come from many reasons, but one of those is people's lack of awareness of wearing masks. Therefore, object detection can be applied to develop models that can detect incorrect mask wearers automatically, helping the government to monitor and reduce the possibility of disease transmission. This work takes advantage of two famous deep learning algorithms in object detection problems, Faster R-CNN and YOLO, with focus on improving the accuracy and detection speed towards real-time detection. The result shows that both 2 models achieve high performance, with YOLOv5 achieving the highest accuracy of 82.5 mAP. The YOLOv5 is better than Faster R-CNN in every term of metrics, from accuracy to speed.

1. Introduction

It has been 2 years since the first outbreak of the COVID-19 epidemic in Wuhan, China, causing more than 5 million deaths and severely affecting the world's economy. Wearing a mask, which is an important preventative action for COVID-19, has become an obligation nowadays. In VietNam, any violation related to not wearing a mask will be fined from 1 to 3 million VND. However, there are still many people who do not comply with this regulation. In fact, there are still many disobediences to the rule, which are a dangerous source of infection and causes great danger to the community. The objective of this project is to apply some object detection model to predict whether a person wears a mask, does not wear a mask, or wears masks in an incorrect way. These object detection models would help generate alerts or reminders if there is any violation to the rule of wearing a mask in a fast and automatic way. Such a system will help raise people's awareness, reduce the cost of human resources to control and repel the epidemic.

2. Data

2.1. Data format and exploration:

- Data source: Kaggle
- 853 images belongs to 3 classes: with mask, without mask and mask weared incorrect
- Each image has a corresponding annotation file which contains information about:
    - Bounding box coordinate in PASCAL format: [xmin, ymin, xmax, ymax]
    - Class label for each bounding box

After exploration, I found out the foreground classes are highly imbalanced, as shown in Fig 1. The ratio between “with mask" and “mask weared incorrect" is 26:1, between “with mask" and “without mask" is 4:1. This will create high bias in the future model. To tackle this problem, I would use a special technique called Weighted Random Sampling, where the probability of an element being selected is based on its weight. In our cases, the weight of each image is determined by the frequencies of “mask weared incorrect" and “without mask" classes.

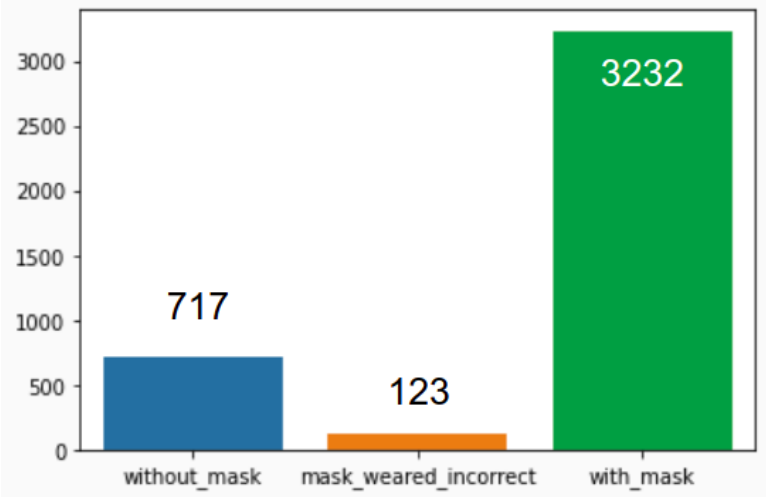
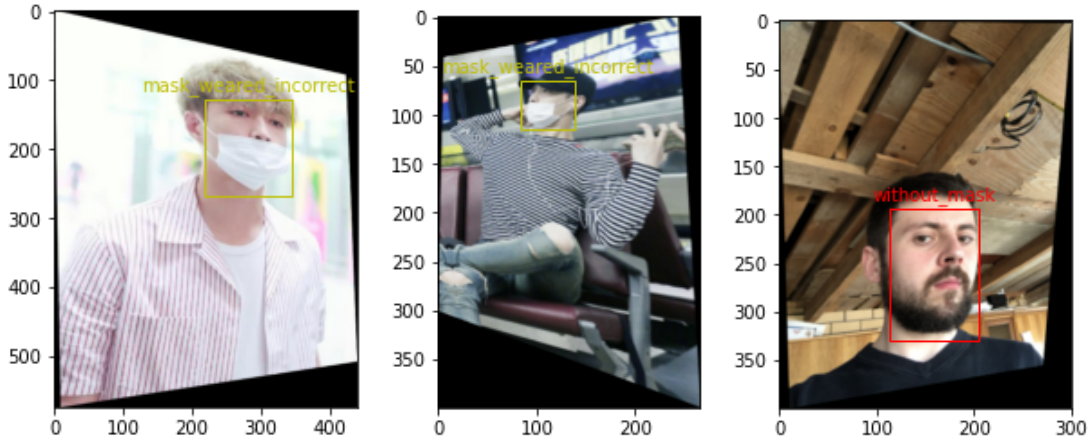


Fig1. Class distribution in Face mask dataset

2.2. Augmentation techniques:

As Weighted Random Sampling is implemented, data augmentation must be included to increase the variability of the input images, so one image can be diversely sampled in different versions and the designed object detection model has higher robustness to the images obtained from different environments. The augmentation methods that I used in this project include:

- Photometric Distortion: Randomly changing the brightness, contrast, saturation, and noise of the image. This will make our model more robust in different lightning conditions.
- Geometric distortion: Random rotate, scale, shear the image to diversify the viewing angle



- Grid Mask [Paper]: Regions of the image are hidden in a grid, this forces our model to learn component parts of what makes up an individual object
    - Extensive experiments show that this method outperforms the latest AutoAugment,which is way more computationally expensive due to the use of reinforcement learning to find the best policies. On the ImageNet dataset for recognition, COCO2017 object detection, and on Cityscapes dataset for semantic segmentation, GridMask improves performance over baselines.

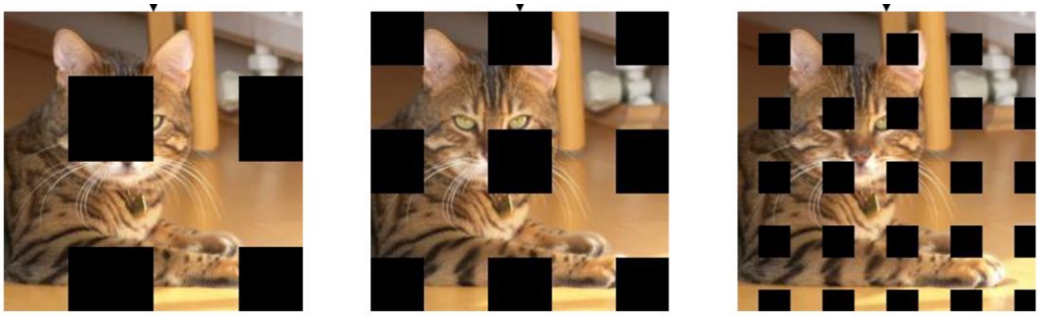


Fig 2. Illustration of Grid Mask

2.3. Training and validation dataset:

For evaluating the model, I will split the original data to a training and validation set with the ratio of 80:20. One point worth noting is Weighted Random Transform and augmentation techniques are only applied on training dataset. The class distribution of the final dataset is shown in Fig 3. The class distribution is more balance with the ratio of 4:1 between “with mask” and “mask weared incorrect"

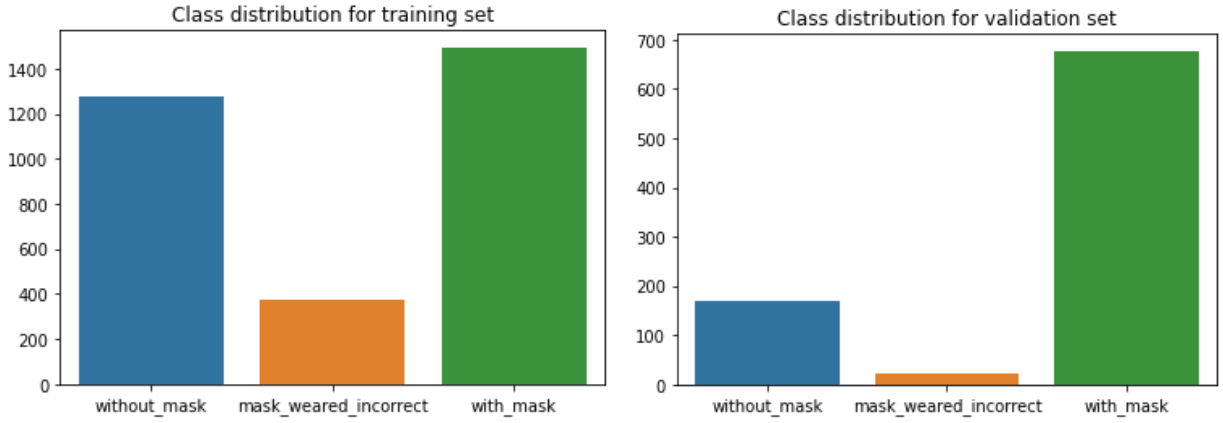


Fig3. Class distribution on training and validation dataset.

3. Methodology

Today there are many famous algorithms successfully solving the problem of object detection, most of them are categorized into two groups: two-stage and one-stage methods. Two stage methods aim to decrease the large number of negative examples resulting from the sliding windows, called anchors, to a manageable size by using a proposal mechanism which determines the regions where the objects most likely appear, called Region of Interests (RoIs). On the other hand, one-stage methods, including SSD Variants, YOLO variants and RetinaNet, are designed to predict the detection results directly from anchors – without any proposal elimination stage – after extracting the features from the input image.

In the scope of this project, I would choose 2 algorithms that are the most well-known and representative for 2 groups, Faster R-CNN for two-stage methods and YOLOv5 for one-stage.

3.1. Faster R-CNN:

The R-CNN model family: The family of methods refers to the R-CNN, which stands for “Regions with CNN Features” was first described in the 2014 paper by Ross Girshick, et al. from UC Berkeley. R-CNN’s have proved highly effective in detecting and classifying objects in natural images, achieving state-of-the-art results on VOC-2012 and the 200-class ILSVRC-2013 object detection dataset.

1. R-CNN (Girshick et al. 2014)
2. Fast R-CNN (Girshick 2015)
3. Faster R-CNN (Shaoqing Ren, et al 2015)

3.1.1. Faster R-CNN Algorithm

Using neural networks, a R-CNN have two 2 main tasks while detecting the object:

- Identify promising region that are likely to contain objects (Region of Interest - ROI)
- Compute the object class probability distribution on each ROI (i.e. the probability that ROI contains an object from a certain class)

To perform those tasks, R-CNNs are designed to have three main types of networks:

1. **Head**: The first few layers of the network learn to detect general features such as edges and color blobs,.. often initialized from a pre-trained network. These layers will identify promising features for further exploitation.
2. **Region Proposal Network**: Take feature maps produced by the head network and use a series of convolutional and fully connected layers to produce promising ROIs that are likely to contain a foreground object. These promising ROIs are then used to crop out corresponding regions from the feature maps produced by the head network, called “Crop Pooling".
3. **Classification Network**: Learn to classify objects contained in each ROI, input taken from crop pooling.

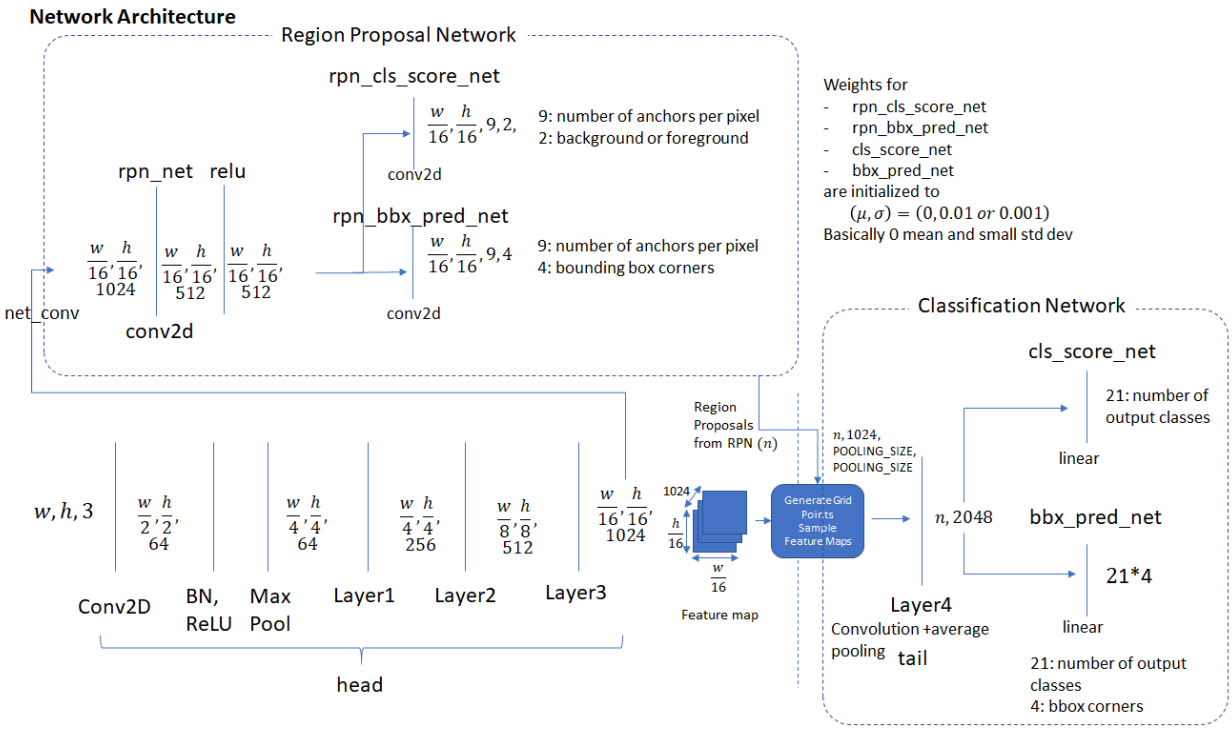
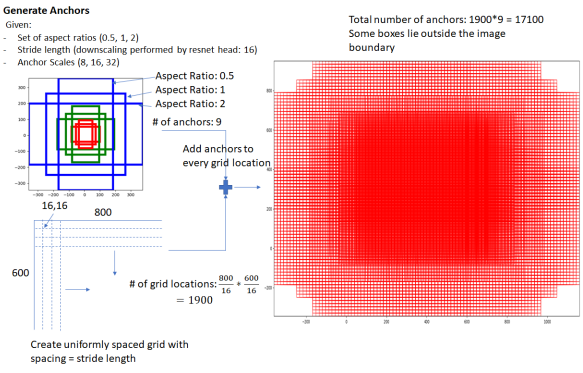


Fig4. R-CNN network architecture.

In the software implementation, R-CNN execution is broken down into several layers, as shown in Fig6. A layer encapsulates a sequence of logical steps that can involve running data through one of the neural networks and other steps such as comparing overlap between bounding boxes, pooling...I will briefly explain the functionality and purpose of each layer.

Anchor Generation Layer: Generate a fixed number of bounding boxes (anchors) by first generating 9 anchors of different scales and then replicating these by moving across equally spaced points over the input image.



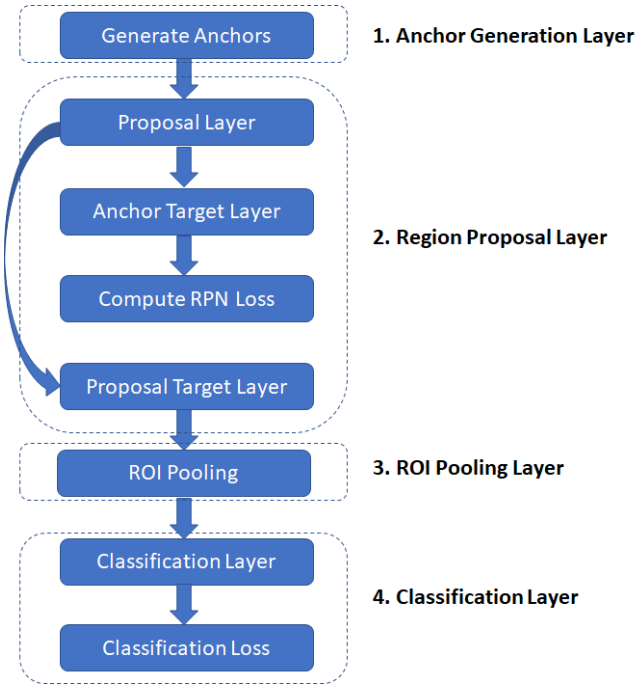


Fig5. Anchor Generation Layer Fig6. Software implementation of Faster R-CNN

Region Proposal Layer: In Faster R-CNN, a "sliding window" based technique is used to generate a set of dense candidate regions. The region proposal layer has two goals:

- Identify background and foreground anchors
- Improve the quality of the anchors (by translating, scaling,..)

ROI Pooling Layer: Using bounding box coordinates from promising ROIs produced by proposal target layer to extract regions from the convolutional feature maps produced by the head network. The extracted feature maps are then run through the rest of the network to produce object class probability distribution and regression coefficients for each ROI.

Classification Layer: The result from the pooling layer is a one-dimensional feature vector for each ROI. The feature vector is pass through 2 fully connected layers: one for predicting class and class probability and one for bounding box

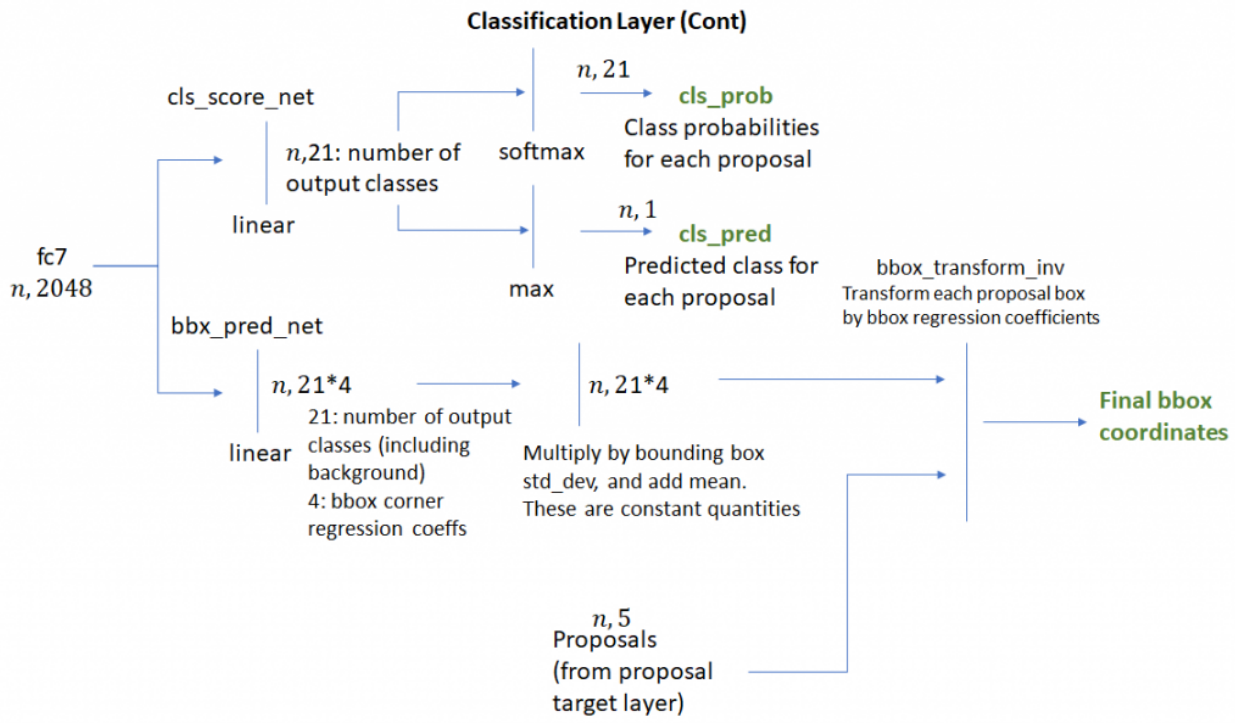


Fig 7. Classification layer

For detection phases, anchor target and proposal target layers are not used . The proposal layer simply applies the bounding box coefficients to the top ranking anchor boxes and performs NMS

(non-maximum suppression) to eliminate boxes with a large amount of overlap. The steps in detection phases are shown in Fig 8.

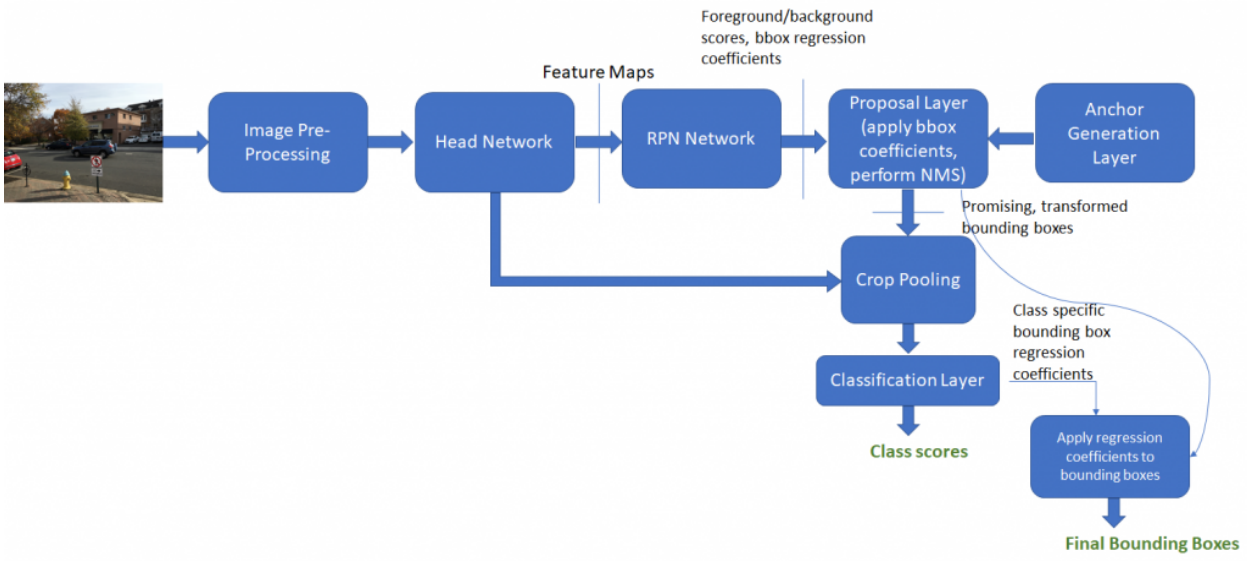


Fig 8. Detection process in Faster R-CNN model

3.1.2. Faster R-CNN implementation

For the problem of face mask detection, I decided to use a pretrained Faster R-CNN model with a ResNet-50-FPN backbone, which was implemented in Pytorch library. Compared to MobileNetV3-Large backbone, the ResNet-50-FPN achieves higher accuracy of 37 boxAP (on COCO val2017), but has twice as much training time. This pretrained model would bring feasible results while saving training time and resources.

Training parameters:

- Number of epoch: 30
- Optimizer: Stochastic Gradient Descent
- Learning rate = 0.005
- Batch size: 4

3.2 YOLO v5

YOLO is an abbreviation for the term “You Only Look Once”. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

3.2.1. YOLOv5 algorithm

The YOLOv5 network consists of three main components:

1. Backbone: A convolutional neural network that aggregates and forms image features at different granularities. Both YOLOv4 and YOLOv5 implement the CSP (Cross Stage Partial Networks) Bottleneck to formulate image features. The CSP are based on DenseNet
2. Neck: A series of layers to mix and combine image features to pass them forward to prediction. Both YOLOv4 and YOLOv5 implement the PA-NET neck for feature aggregation.
3. Head: Consumes features from the neck and making prediction of boxes

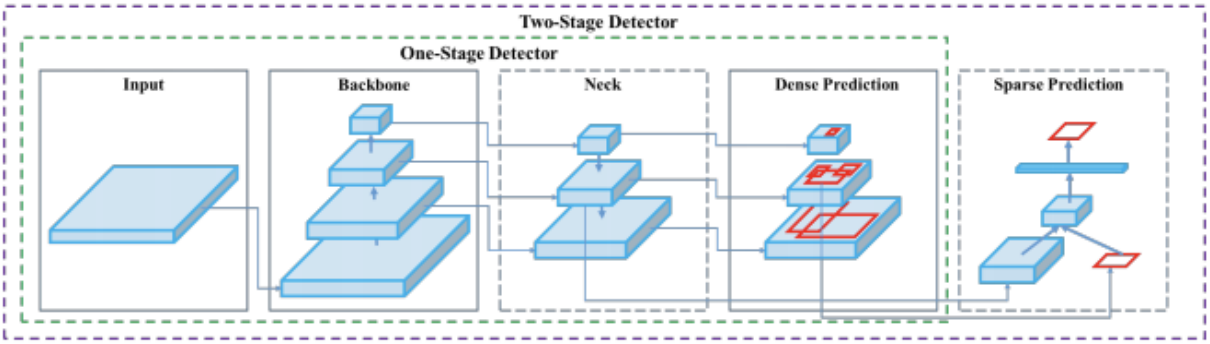


Fig 9. Object detector

Loss function for training: Sum from GIoU, object and class losses.

For enhance the performance, YOLOv5 uses other techniques such as: data augmentation (Mosaic, CutOut,...) and Auto Learning Bounding Box Anchors

3.2.2. YOLO v5 implementation

Model	AP^{val}	AP^{test}	AP_{50}	$Speed_{GPU}$	FPS_{GPU}		params	FLOPS
YOLOv5s	36.6	36.6	55.8	**2.1ms**	**476**		7.5M	13.2B
YOLOv5m	43.4	43.4	62.4	3.0ms	333		21.8M	39.4B
YOLOv5l	46.6	46.7	65.4	3.9ms	256		47.8M	88.1B
YOLOv5x	**48.4**	**48.4**	**66.9**	6.1ms	164		89.0M	166.4B
YOLOv3-SPP	45.6	45.5	65.2	4.5ms	222		63.0M	118.0B

Table 1. Different model of YOLOv5 (from YOLOv5 repo)

For the implementation of YOLO v5 with this problem, I choose the YOLOv5x with pre-trained weight since it has the highest accuracy among those variants.

Training parameters

- Number of epochs: 30
- Image size: 640
- Batch size: 4
- Optimizer: Stochastic Gradient Descent

4. Result

The evaluation is done using the standard Mean Average Precision (mAP) at some specific IoU threshold (mAP@.5 and mAP@.5:.95). mAP is a metric that comes from information retrieval, and is commonly used for calculating the error in ranking problems and for evaluating object detection problems.

	Faster R-CNN			YOLO v5		
Class	mAP@.5	mAP@.5:.95	FPS	mAP@.5	mAP@.5:.95	FPS
without_mask	86.8	52.4	6.36	89.6	53.6	6.77
mask_weared_incorrect	65.7	35.9		61.5	39	
with_mask	89.6	59		96.3	64.6	
All	80.7	49.1		82.5	52.4	

Table 2: Evaluation result from 2 models

In accuracy of each class, both models have good predictions on “without mask" and “with mask" with over 85 mAP@0.5, whereas “mask weared incorrect" have lower points on both models. This can be explained by the lack of data on this class.

Comparison between Faster R-CNN and YOLOv5: Both 2 models achieve high performance, with YOLOv5 achieving the highest accuracy of 82.5 mAP. The YOLOv5 is better than Faster R-CNN in every term of metrics, from accuracy to speed.

4.1. Faster R-CNN result

Learning progress from Faster R-CNN: The mAP point of “without mask” and “with mask" are very good from the first epoch, this can be explained since the model are pre trained on COCO, so it has some information on detecting people's faces.

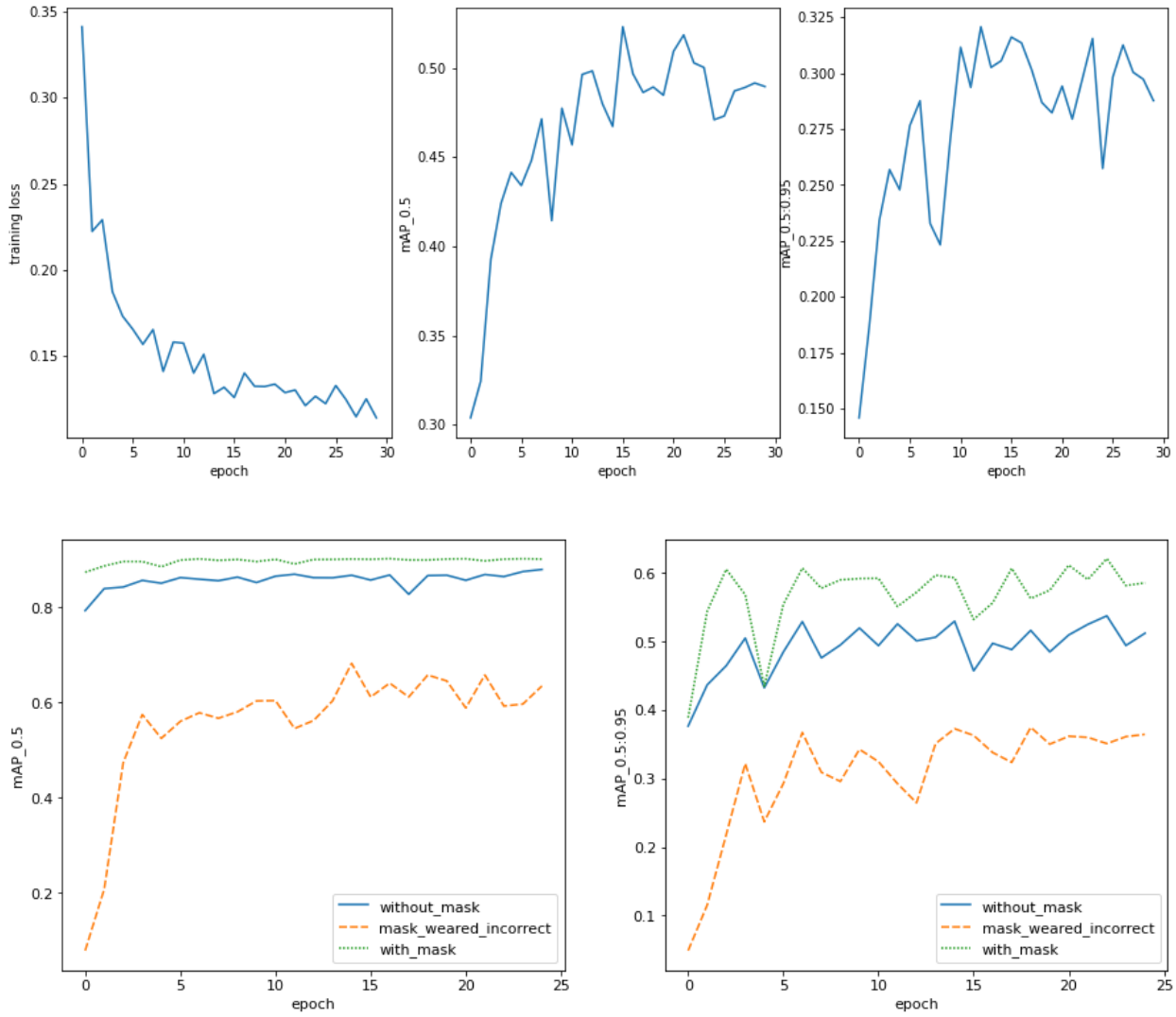


Fig 10. Learning progress on Faster R-CNN

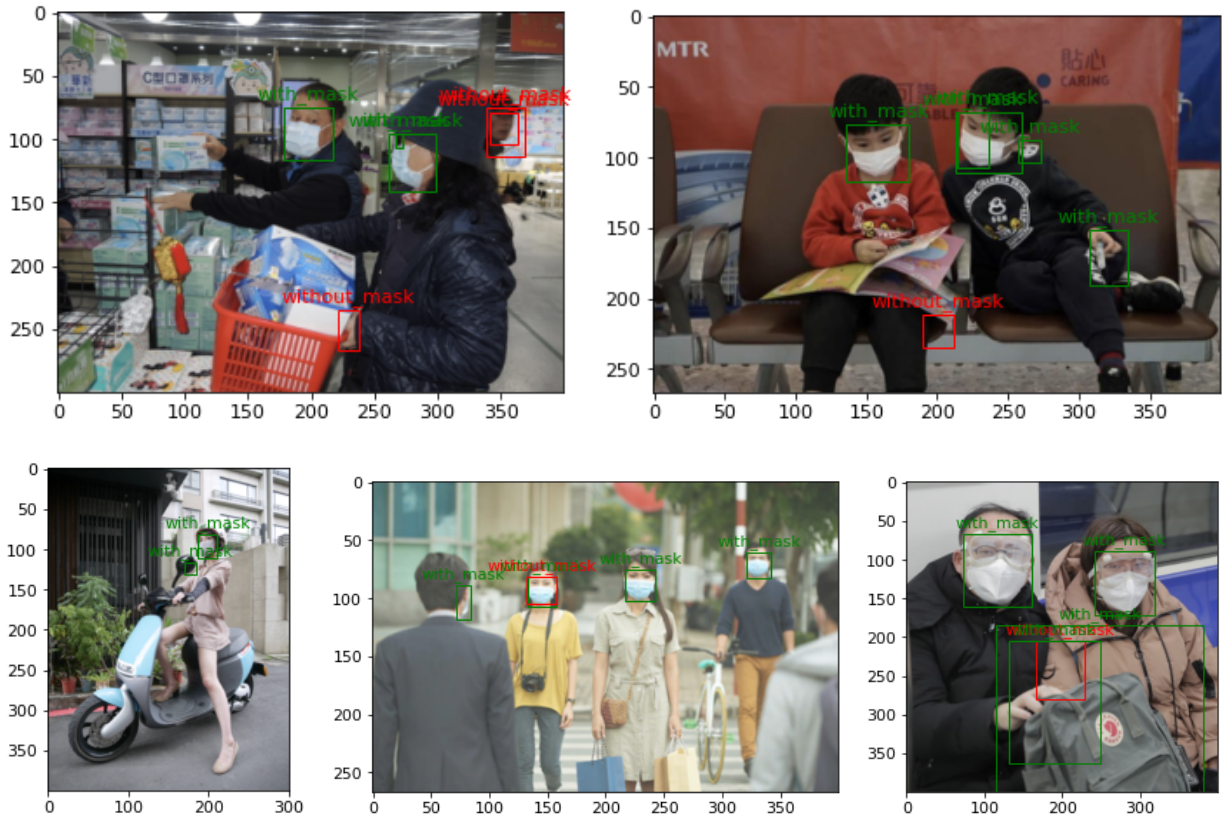


Fig 11. Detection result on validation set of Faster R-CNN

Although the model has high mAP points, we observed a lot of overlapping and misdetection from Faster R-CNN.

4.2. YOLO v5 results:

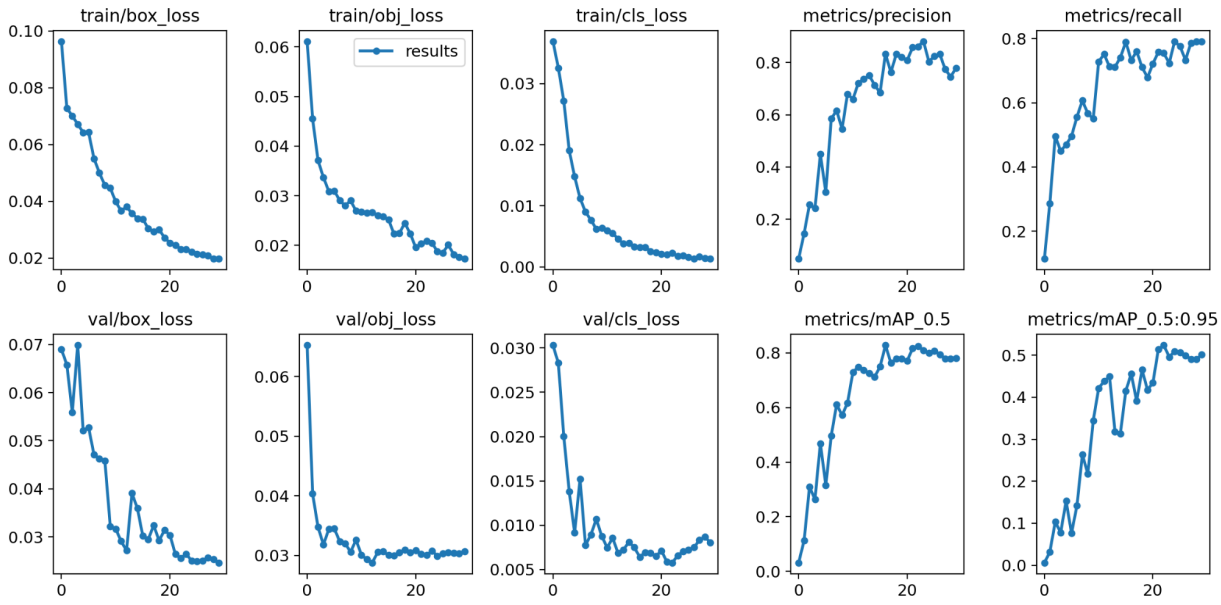


Fig 12. Learning progress of YOLOv5

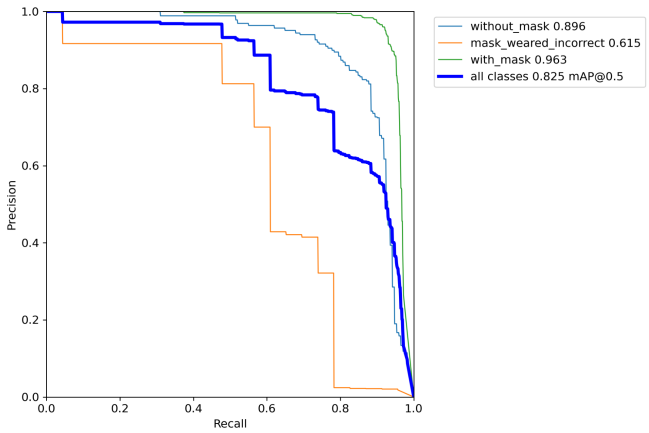


Fig 13. Precision - Recall curve from YOLO v5

Fig 14. Detection result on validation set of YOLOv5

5. Conclusion and future work:

Conclusion:

- YOLO perform quite well with test set, less overlapping instance and accurate
- Faster R-CNN have quite good mAP, but since need a lots improvement since the prediction contains a lots overlapping instances, some time misdetected class

Future work:

- Collect more data for “mask weared incorrect” class, vary the data in different face posture, lightning condition and crowds.
- Improve Faster R-CNN by trying different backbones.
- Try more variants of YOLOv5 to increase the FPS of the model.