Real time facial mask detection

Trent Zhang, Xingzhong Fan

November 5, 2023

Abstract

In this project, we construct a back end program to detect facial mask in both static images and webcam video stream. There is two main part in this project. First, we explore models to recognize face bounding box in a given image/video, and crop the face from the video frame. Second is to detect whether the face in the cropped frame properly wore a facial mask or not.

For the first part, we plan to use MTCNN to recognize the face in video frame. For the second part, we implement multiple algorithms and models to detect facial mask on Face Mask Detection.

1 Introduction

In this project, we will construct a back end program to detect facial mask in video stream. There is two main part in this project. First, we need to construct a model to recognize face in a given video frame, and crop the face from the recognized result. Secondly, we will detect whether the face in the cropped frame properly wore a facial mask or not.

For the first part, we plan to use MTCNN[8] to recognize the face in video frame. For the second part, we will implement multiple deep learning algorithms and models to detect facial mask on Face Mask Detection Data-set from Kaggle.

2 Member roles

Trent Zhang

Implement, construct a model to detect and crop face. Construct 2 mask detection pipelines for both static image and webcam video stream.

Xingzhong Fan

Implement, construct and compare different Deep Learning models to detect facial masks.

3 Face Detection

In order to detect facial masks with given static image or video stream, we first need to construct a model to detect faces. If we are planing to construct a real-time face mask

recognition system, so our data will be a video stream. But in order to test our method, it's better to start with experimenting on static images, so we used some images from web, one example[1] is as Figure 1.

Figure 1: Example: People on street during Covid-19

3.1 MTCNN

In this part, we introduce the structure, working principle and application of MTCNN used for face detection, alignment, cropping and scaling. We will give a brief introduction to the advantages of the MTCNN algorithm and the structure of its sub-networks.

The MTCNN algorithm is an algorithm based on deep learning to realize face detection and face alignment. This algorithm can complete the two tasks of face detection and alignment at the same time. Compared with traditional face detection and alignment algorithms, it has the highest operating efficiency. The MTCNN algorithm contains three sub-networks that process face images from rough to detailed: Proposal Network (PNet), Refine Network (RNet), and Output Network (ONet).

Besides, since MTCNN model is selecting final face bounding box from lots of potential face boxes, the computation cost of it will be high. So in order to speed up our pipeline, we downsized our input images to get a faster computation.

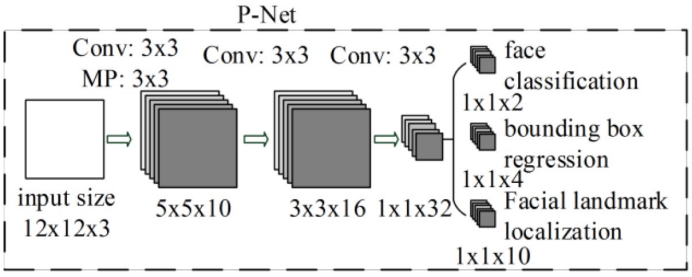

Figure 2: P-Net Structure

[1]This image is acquired from www.thedp.com

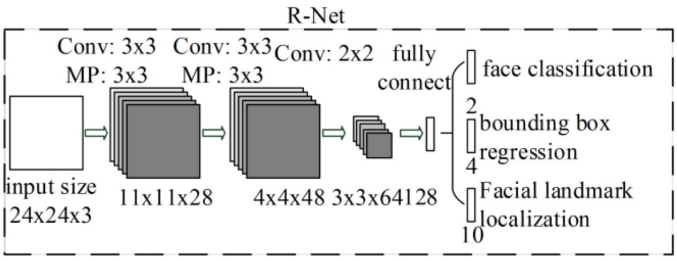

Figure 3: R-Net Structure

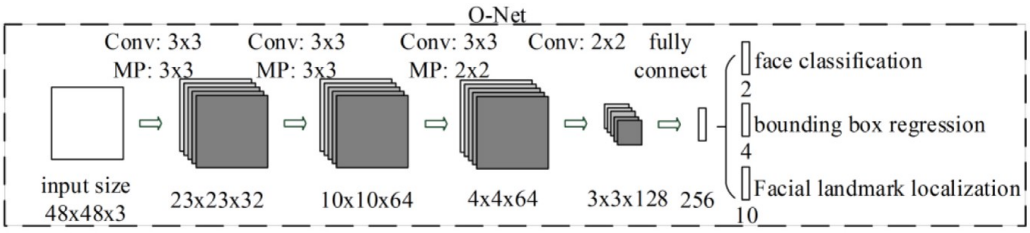

Figure 4: O-Net Structure

3.2 Result

After implementing the MTCNN face detection method, we successfully detected the faces in the image, and the result of Figure 1 is as Figure 5. In order to comply with the face mask detection part, the cropped faces are resized to 160×160 pixels, the final result of the top 4 confident faces are presented in Figure 6.

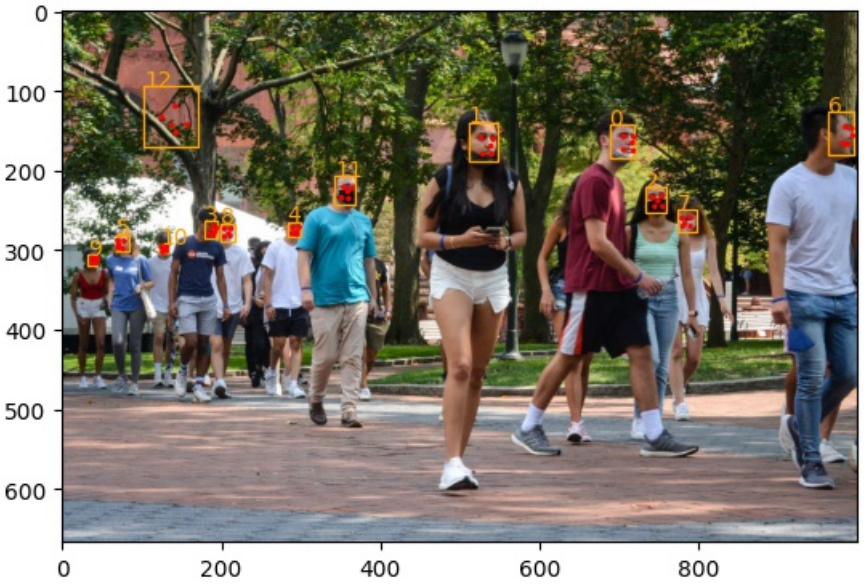

Figure 5: Example Result: People on street during Covid-19

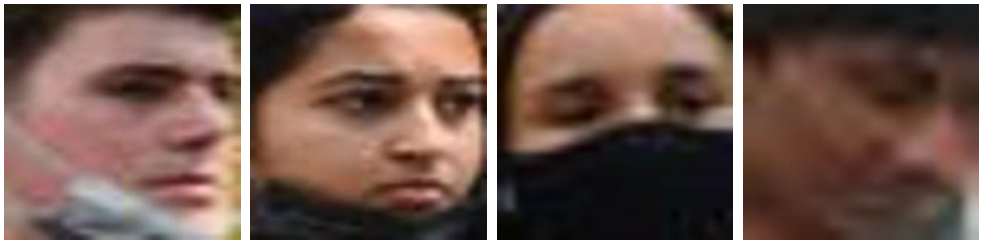

Figure 6: Example top 4 faces: People on street during Covid-19

4 Mask Detection

In this part, we use Keras to build computer vision deep learning algorithms, such as convolutional neural networks. In addition, we also use some neural network architectures and compare their ability to recognize masks. The evaluation index is the accuracy of the test set. In addition, we also record the time taken for each epoch of each model. All experiments are run on Google Colab with Tesla P100 GPU.

4.1 Methods

Training set has 8082 images, Testing set has 900 images, and these data are divided into three categories: mask_weared_incorrect, with_mask, without_mask, and the data is shuffled. In addition, when all the following models are running, we set up a method of adaptive learning rate. When the accuracy of the test set does not increase by more than 3 epochs, the learning rate becomes half of the previous one. In addition, we also set an early stopping strategy, patience is 5.

4.1.1 CNNs

we use keras to implement three Convolutional Neural Networks (CNNs)[4] for mask recognition. The first model is composed of convolutional layer, pooling layer, dropout, convolutional layer, pooling layer, dropout, flatten, fully connected layer, dropout, fully connected layer, fully connected layer.The second model consists of convolutional layer, pooling layer, dropout, convolutional layer, dropout, pooling layer, convolutional layer, dropout, pooling layer, flatten, and two fully connected layers. The third model consists of convolutional layer, convolutional layer, dropout, pooling layer, convolutional layer, dropout, convolutional layer, dropout, pooling layer, flatten, and two fully connected layers. The accuracy and loss of the model are shown in figure 7, from left to right are model1, model2 and model3. Model1 performed best on the test set, with a accuracy of 0.86.

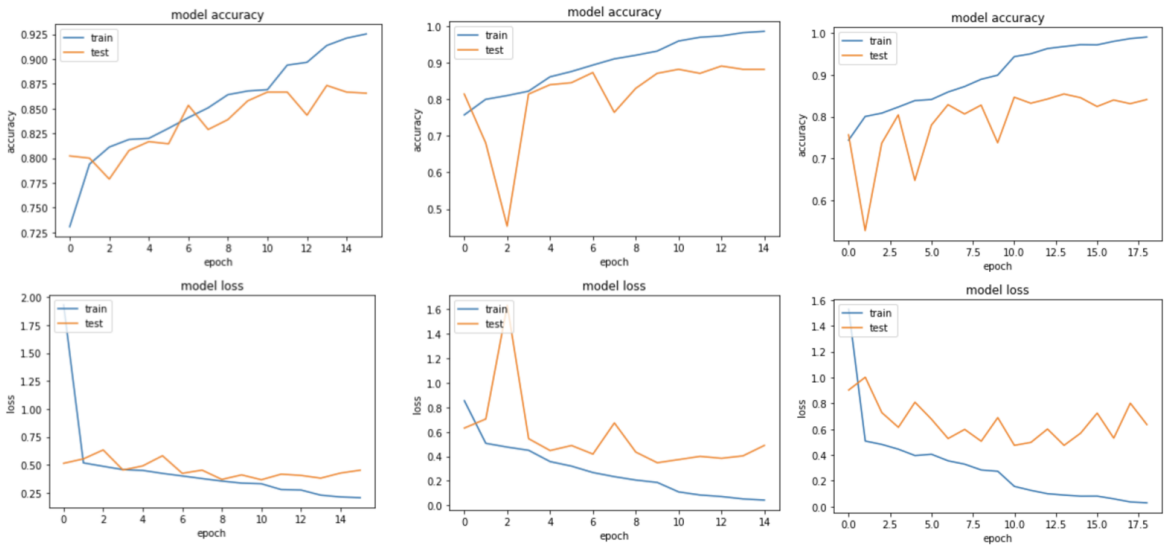

Figure 7: Different CNNs Model accuracy and loss

4.1.2 ResNets

we try to use different residual neural network architectures [1]. ResNet is to ease the training of networks that are substantially deeper than those used previously. ResNet has added a residual learning unit to the neural network, because residual learning is easier than direct learning of original features. When the residual is 0, the stacked layer only does the identity mapping at this time, at least the network performance will not decrease, in fact the residual will not be 0, which will also make the stacked layer learn new features based on the input features , So as to have better performance.

we used ResNet50, 101, and 152 respectively. The difference between them is the number of parameters, which are 25,636,712, 44,707,176, 60,419,944. Therefore, the training time is also increasing. The time of each epoch is 20s, 32s, and 46s. The accuracy rate on the test set is 0.92, 0.83, 0.9622. The final decrease in accuracy on the ResNet101 training and test set was due to the use of adaptive learning rate which led to the final decrease. In general, the deeper the residual nerve, the better the performance on the test set.

In addition, we also tried ResNetV2 [2], which uses a new residual unit, and the author suggests that the forward and backward signals can be directly propagated from one block to any other block when using identity mappings as the skip connections and after-addition activation. This residual neural network has higher accuracy and faster speed.

In my experiment, we used ResNet50V2 and ResNet152V2. The difference between them lies in the number of parameters, which are 25,613,800 and 60,380,648, respectively. Compared with the first-generation residual neural network, their number of parameters is relatively small, so the training time for each epoch is also less, which is 18s and 45s, respectively, which is 2s and 1s faster. However, the accuracy rate on the test set has risen to 0.9567 and 0.98 respectively. ResNet50V2 has risen extremely significantly, increasing by 4%, and its accuracy is almost the same as that of the first-

generation network ResNet152, but the training time is one-third of the former. The accuracy and loss of the model are shown in figure 8, first row from left to right are ResNet50, ResNet101 and ResNet102, second row from left to right are ResNet50V2 and ResNet152V2. ResNet152V2 performed best on the test set, with a accuracy of 0.98.

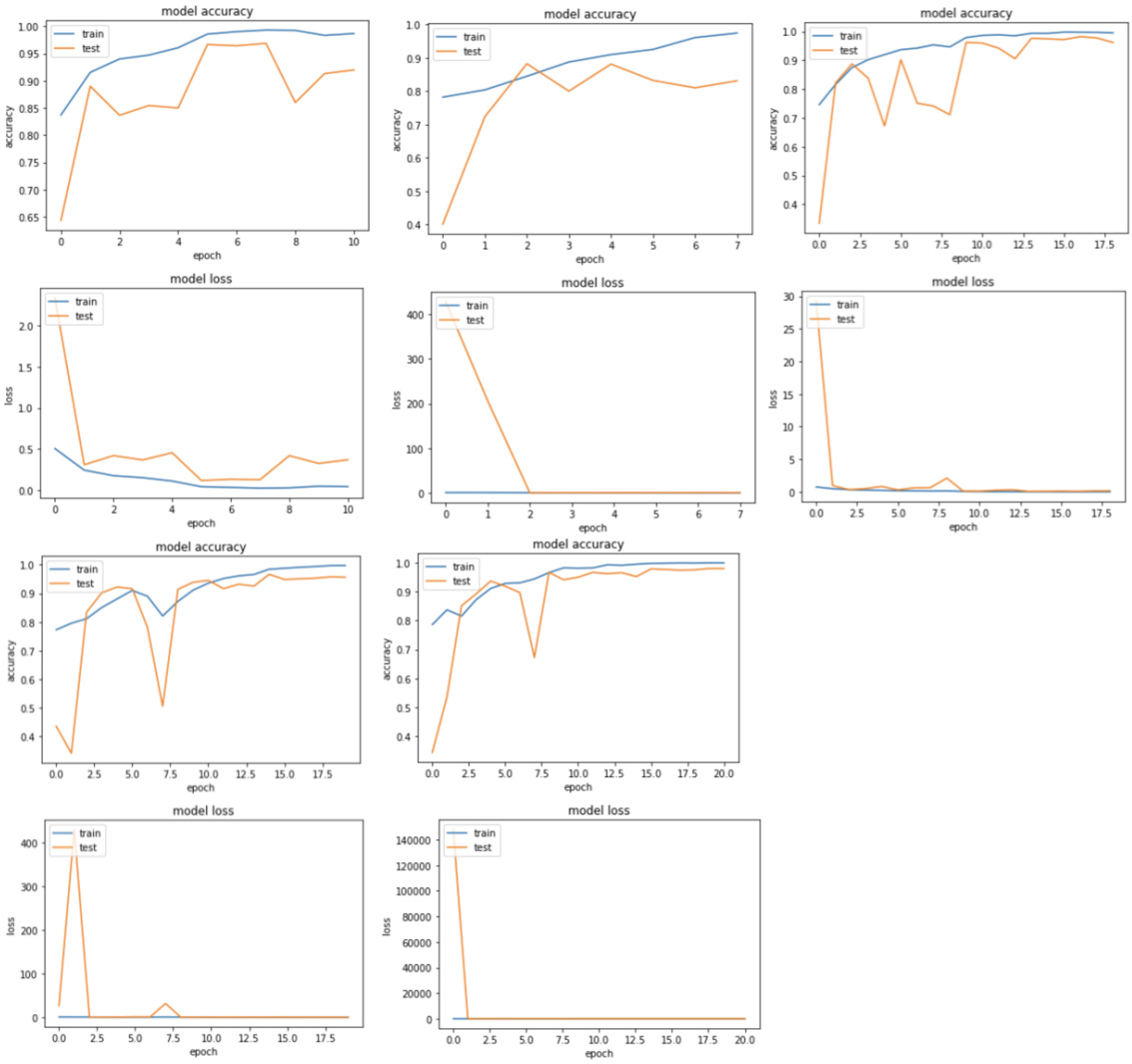

Figure 8: Different ResNets Model accuracy and loss

4.1.3 InceptionNets

we use Inception Architecture [6] in this part. This method uses the method of exploring extended networks, which aims to use the increased calculations as efficiently as possible through proper decomposed convolution and active regularization. This method can achieve very good performance with relatively low computational cost. On this basis, we also tried to use the Inception-ResNet architecture [5], and introduced residuals on the basis of the Inception Architecture.

The accuracy of InceptionV3 on the test set is as high as 0.97, and it takes only

19s in each epoch. In addition, the accuracy of using IncepotionResNet is 0.9911, and the time used is 41s. In the mask recognition project, the accuracy of InceptionNet is already higher than that of residual neural network, and in addition, it takes less time than residual neural network. The accuracy and loss of the model are shown in figure 9, left is InceptionV3 and right is IncepotionResNet.

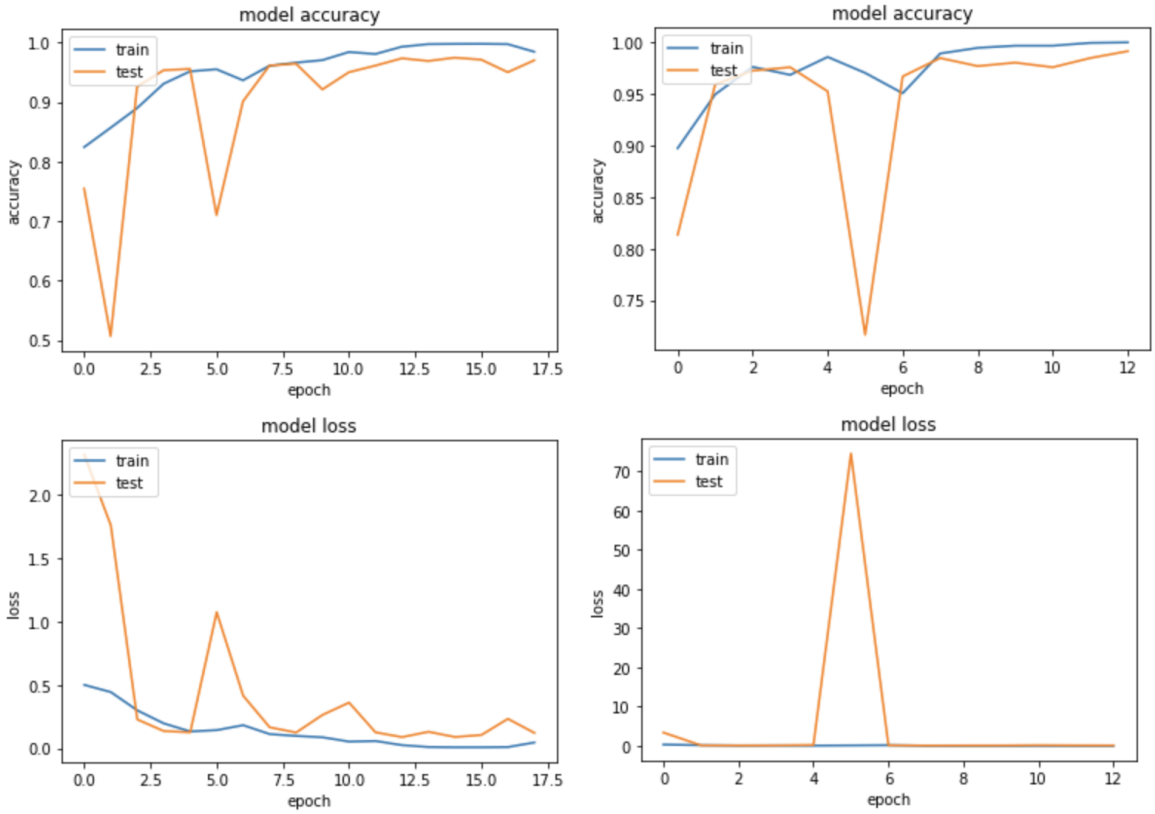

Figure 9: Different InceptionNets Model accuracy and loss

4.1.4 DenseNets

For this part, we use Densely Connected Convolutional Networks [3]. If convolutional networks contain shorter connections between the layer close to the input and the layer close to the output, they can be trained more deeply, more accurately, and more effectively. The DenseNet feed-forward method connects each layer to each other layer. The feature maps of all previous layers are used as input, and its own feature maps are used as the input of all subsequent layers. DenseNets can alleviate the problem of gradient disappearance, strengthen feature propagation, encourage feature reuse, and greatly reduce the number of parameters.

we used the DenseNet121 and DenseNet201 architectures in keras. The difference between them is the number of parameters and the depth. The first depth is 121 with 8,062,504 parameters, and the second is 201 with 20,242,984 parameters. The first training time is 23s, and the second is 25s per epoch. The accuracy on the test set is 0.9867 and 0.9822 respectively. On the contrary, the accuracy of the first relatively

shallow architecture is higher. we guess it is because of the deeper depth that there may be some overfitting. The accuracy and loss of the model are shown in figure 10, left is DenseNet121 and right is DenseNet201.

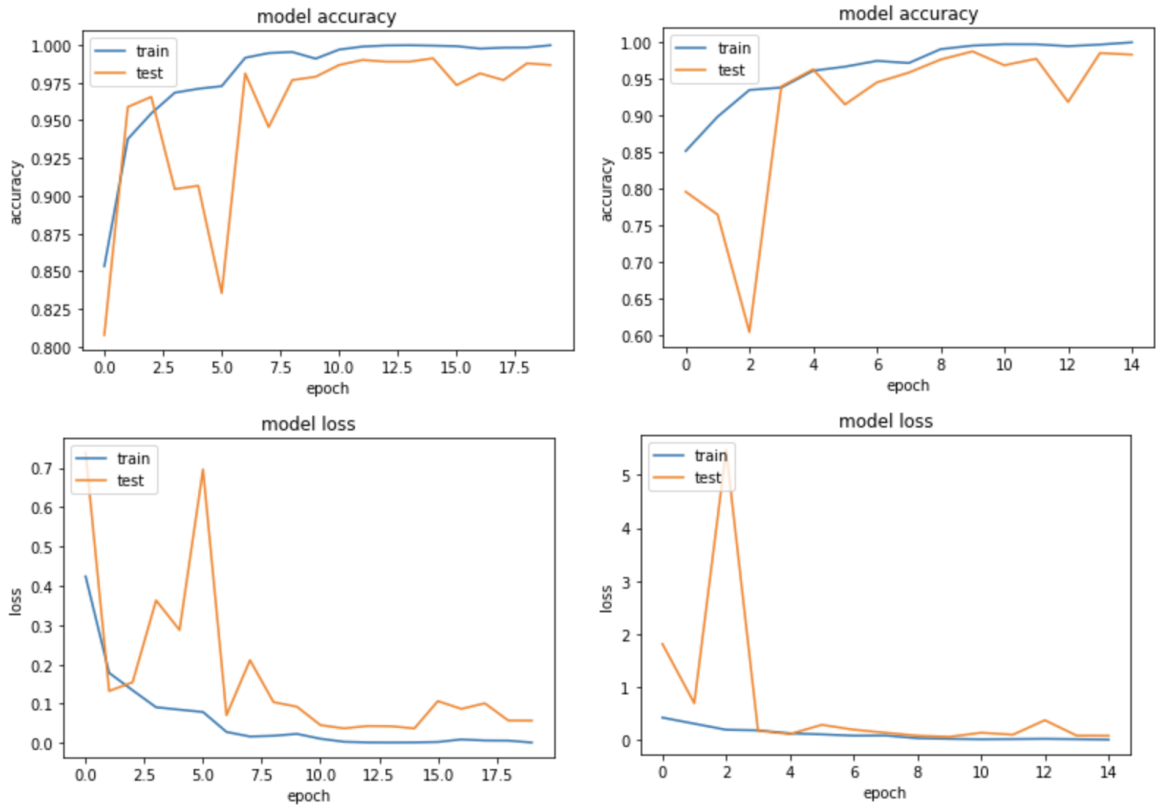

Figure 10: Different DenseNets Model accuracy and loss

4.1.5 EfficientNets

In the last part, we used EfficientNet [7], which is based on neural architecture search and the new model scaling method, which uses simple and efficient composite coefficients to uniformly scale all dimensions of depth/width/resolution. This methods can be much better accuracy and efficiency than previous ConvNets.

we used EfficientNetB0, EfficientNetB3, and EfficientNetB7 respectively. The difference lies in the number of parameters, which are 5,330,571, 12,320,535, 66,658,687 respectively. EfficientNetB7 has the most parameters among all tried deep learning architectures. It takes 106s to train one epoch, but the performance is not very good, with an accuracy rate of 0.9789. In contrast, the accuracy rates of EfficientNetB0 and EfficientNetB3 are 0.9844 and 0.9944. Among them, the performance of EfficientNetB3 on the test set is the best of all models.The accuracy and loss of the model are shown in figure 11, ,from left to right is EfficientNetB0, B3 and B7.

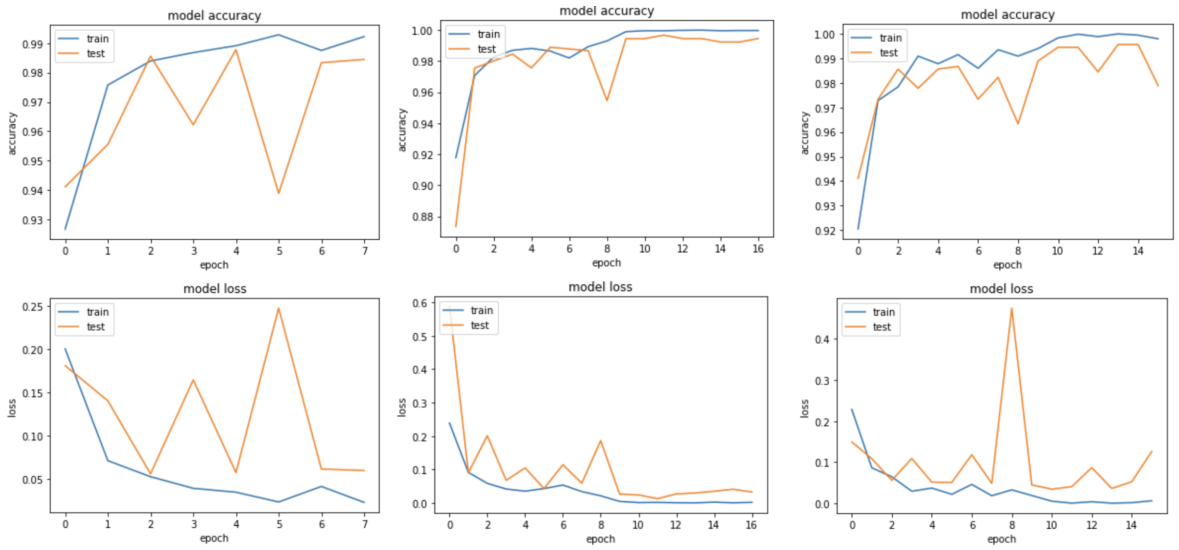

Figure 11: Different EfficientNets Model accuracy and loss

4.2 Comparison and Discussion

In the mask recognition project, we tried to implement different deep learning models and compare 3 CNNs, different ResNets, InceptionNets, DenseNets and EfficientNets. Among them, EfficientNetsB3 performed best, with an accuracy rate of 0.9944 on the test set and 38s to train an epoch. The time required was moderate among the models we tried, so we decided to finally use this deep learning model. The accuracy of all models, test set loss, test set accuracy, and time for each epoch are shown in Table 1.

In this part, we also found some interesting things. The accuracy of using the existing framework is higher than that of the simple convolutional neural network we implemented. In addition, a more complex neural networks tend to take more time, and the performance of networks closer to the present tends to be better. Just like in this project, the accuracy of InceptionResNet is better than that of ResNet, and the accuracy of EfficientNet is higher than that of InceptionResNet. In addition, the performance of networks that are too complex is sometimes not very good, because over-fitting is likely to occur, and the performance on the test set will be inferior to relatively shallow networks.

We randomly selected 8, 8, and 9 pictures respectively without the label of wearing the mask correctly, wearing the mask, and without wearing the mask. The output figure is in figure 12. It can be seen from this picture that all the predictions of the EfficientNetB3 model are correct.

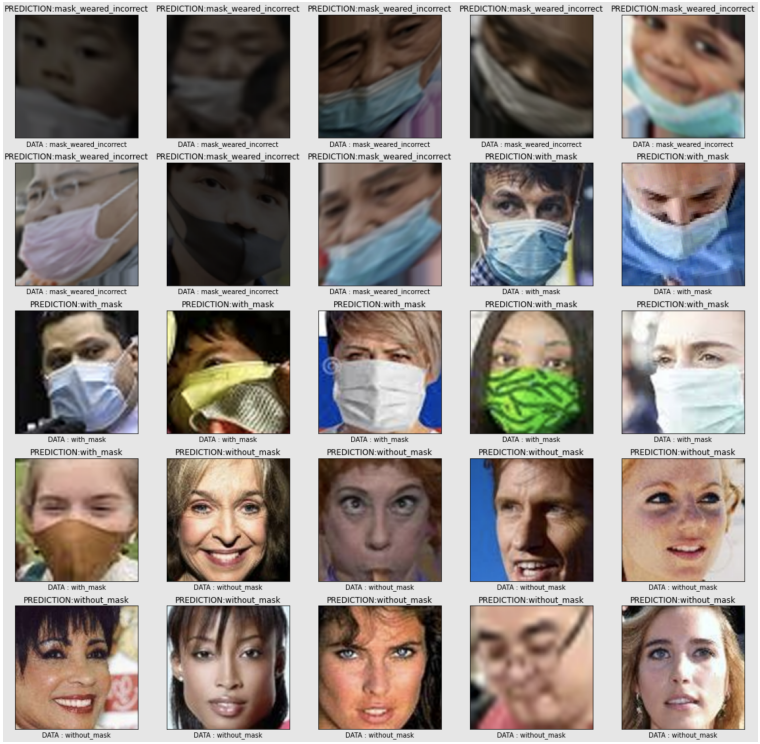

Figure 12: Example EfficientNetB3 Prediction

	train_accuracy	test_loss	test_accuracy	time/epoch(s)
ResNet50	0.9869	0.3676	0.92	20
ResNet101	0.9741	0.4774	0.83	32
ResNet152	0.9957	0.1563	0.9622	46
ResNet50V2	0.9975	0.2154	0.9567	18
ResNet152V2	0.9993	0.1043	0.98	45
InceptionV3	0.9844	0.1209	0.97	19
InceptionResNet	0.9996	0.0938	0.9911	41
DenseNet121	0.9998	0.0572	0.9867	23
DenseNet201	0.999	0.0762	0.9822	35
EffectiveNetB0	0.9922	0.0602	0.9844	22
EffectiveNetB3	09996	0.033	0.9944	38
EffectiveNetB7	0.9979	0.1263	0.9789	106

Table 1: Different Deep Learning Model Output.

5 Final result and analysis

We applied our best mask detection model and constructed a pipeline to the detect facial masks in static images. Here, we use Figure 6 as example, the mask detection result is as Figure 13. From the detection result, we can see that the precision of mask detection is not as accurate as we expected, and the main cause for this is low resolution of the source image.

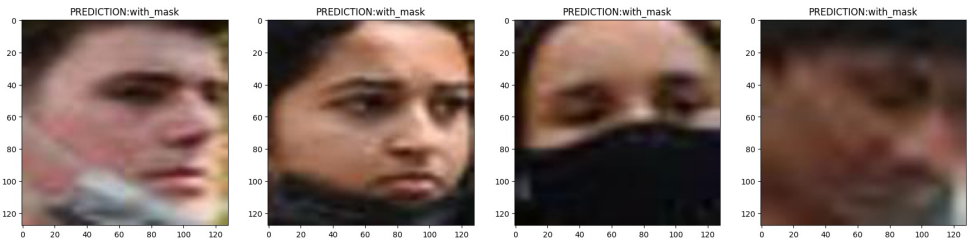

Figure 13: Result top 4 faces

For webcam video stream, we also constructed a pipeline for this specific scenario. The result of our facial mask pipeline is overall function good during our testing. Some notable output frames, with our detection pipeline, during the testing is as 14. From the result, we can see that our facial mask detection model seems to detect all kinds of "mask", not only the surgical mask. For example, when we use our hand to "mask" our face, the model will output that the video frame is with mask. Besides, our facial mask detection model doesn't work well in some scenarios, for instance, when the subject is looking aside, the mask detection model will always output "with mask", which is wrong. A possible explanation for this is that, our mask detection model is actually detecting whether there is a bottom area, which are filled with similar pixels, in the face bounding box. For a half face profile, the bottom area of it is filled with similar pixels which is the skin on face, so our model will output the result as "with mask". In the future, we should dig into that and try to find a better way to detect facial masks.

Figure 14: Result Webcam

References

[1] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[2] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. arXiv: 1603.05027 [cs.CV].

[3] Gao Huang et al. *Densely Connected Convolutional Networks*. 2018. arXiv: 1608.06993 [cs.CV].

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[5] Christian Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: 1602.07261 [cs.CV].

[6] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV].

[7] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[8] Kaipeng Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503.