Name: Man Adrian Ho Hin
SID: 20731801
HKUST email address: ahhman@connect.ust.hk

Project Report (Face Mask Detector)

Dataset & Preprocessing:

In this project, I used the FaceMask Dataset in order to create a model for face mask detection.

The dataset contains 2 sub-dictionaries, which 0 contain the image without mask and 1 contain the image with mask, there are 687 and 691 items respectively. Around half of the images are generated with augment.

For preprocessing, it is better with the same input size for a CNN model. So I first create a data generator with the function tf.keras.preprocessing.image.ImageDataGenerator(), setting the resale parameter to 1/255 to normalize each data point to [0,1], also split the dataset with 80% training set and 20% valid set. Then generate the training and valid data set with the attribute flow_from_directory(), with changing the image size into 32 x 32 from any images, which a such small size aim to reduce the training time. Also setting shuffle to True in order to prevent the overfitting problem.

Task(s):

Build a binary classification model using TensorFlow and Keras for detecting whether people in the image wear face mask or not.

Try to minimize the training time and improve the performance (accuracy).

Try to extent the ResNet model with the inception block idea and compare the result.

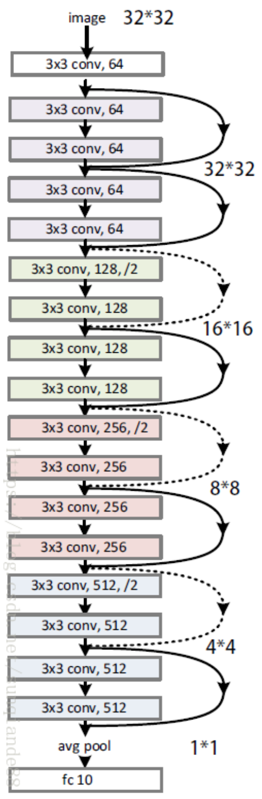
Setting(s):

Hardware Used:	MacBook Pro (13-inch, 2018) Processor: 2.3 GHz Quad-Core Intel Core i5 Memory: 8 GB 2133 MHz LPDDR3 Graphics: Intel Iris Plus Graphics 655 1536 MB
Software Used:	Jupyter Notebook Python version: 3.9.12 Library used: tensorflow (version: 2.11.0) keras (version: 2.11.0) time os

Machine learning methods:	CNN architecture 1. ResNet-18 2. ResInceNet-10 (custom) 3. ResInceNet-18 (custom)	
Parameters:	Number of Classes:	2
	Classes:	0 (without mask) 1 (with mask)
	Image Input Size:	32 x 32
	Batch Size:	32
	Splitting Scale:	training set (80%) valid set (20%)

<u>Description of the experiments</u>

First, I build up a ResNet-18 with referencing to PA2. Basically, this model is formed by 18 layers. The first layer is a convolutional layer with kernel_size=3 and output channel=64, then perform the BatchNormalization, relu activation and max pooling with pool_size=2. The last layer is a fully connected layer and connected to 2 outputs (Classes) after the global average pooling, which perform the classification work. Moreover, the layer in the middle is formed by 4 Res_Block and each Res_Block contain 2 Basic_Block, which determined by the output channel size, 64, 128, 256 and 512 respectively. Turning the images from 32 x 32 to 16 x 16 to 8 x 8 to 4 x 4. The structur



Each Basic_Block start with a identity function used for skip connection. Downsample may be applied if different in image sizes. Then following up by a convolutional layer, BatchNormalization, relu activation, another convolutional layer, another BatchNormalization, skip connection by adding the processing data and the identity, lastly another relu activation.

After setting up all the structures, compile the model with Adam optimizer with 0.00001 learning rate. I set that to 0.0001 initially but the accuracy and val_accuracy usually fluctuating and val_accuracy stop at around 0.5, therefore I choose a smaller learning rate. Also a binary crossentropy loss function as a binary classification problem, although it is workable for training with mse while **Error! Bookmark not defined.**it seems binary crossentropy more appropriate. And a metrics show the BinaryAccuracy.

For the model fitting part, I set the epochs=100 and apply the early stopping with patience=10 according to the mode='max' and val_accuracy. It is because it seems that my model usually has a 0.5 accuracy at the beginning epochs and requires several more epochs to increase the accuracy. However, around up to 20 epochs the accuracy won't have significant improvements.

The second (ResInceNet-10) and third (ResInceNet-10) is based on the ResNet structure, imply the inception block idea in it. I was inspired by the similarity between the Basic_Block and inception model. So I changed the second convolution layer in Basic_Block to a GroupConv, which formed by 1 x 1 and
3 x 3 kernel size convolution layers and adding them together. For sure normalization and relu activation also performed in GroupConv.

The second one's compile is basically the same with the first one, while the compile for the third one changed the epochs=100 and patience=20 as need even more epochs to prevent the early stopping with poor accuracy.

Discussion:

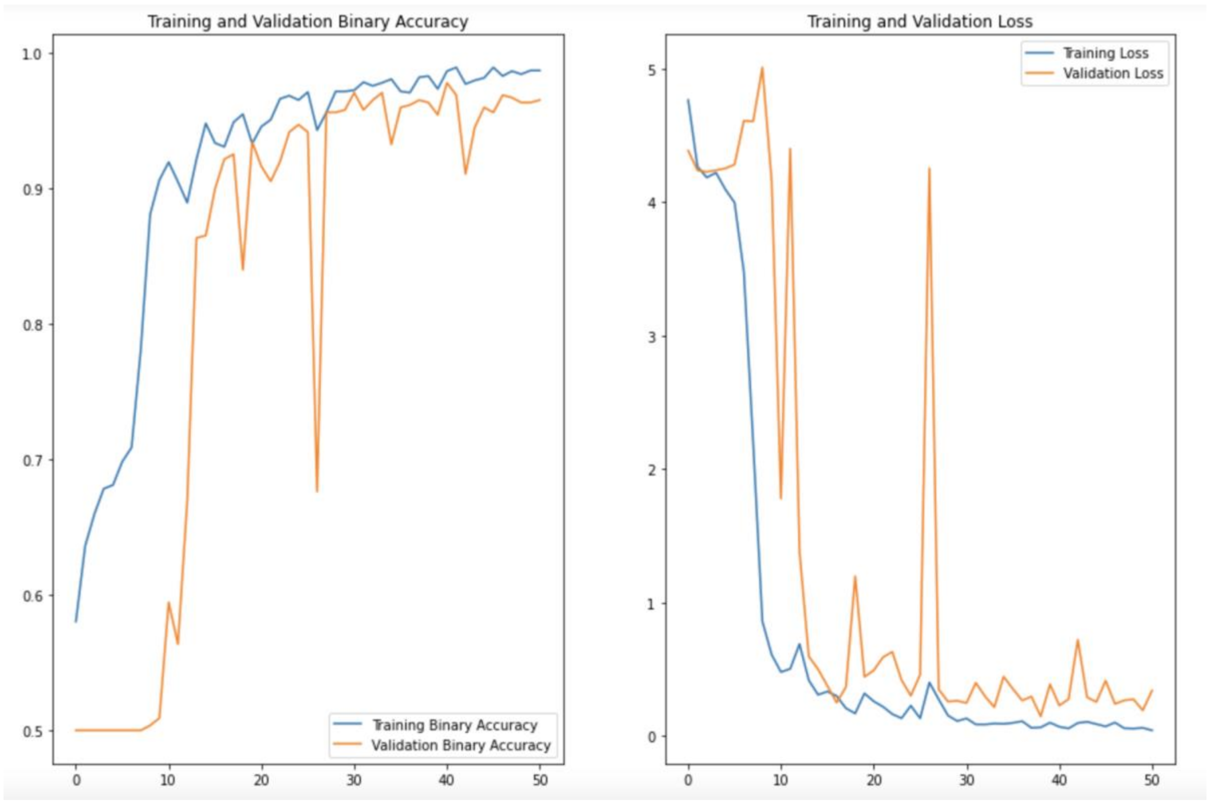
So the param for model 1,2&3 are #11180674, #5260994 and #11894274 respectively.
The training time are 1672s, 1069s and 1537s respectively.
The final performance are

```
#1 loss: 0.0393 - accuracy: 0.9873 - val_loss: 0.3377
- val_accuracy: 0.9655
#2 loss: 3.9119 - accuracy: 0.7393 - val_loss: 4.0341
- val_accuracy: 0.7255
#3 loss: 0.1393 - accuracy: 0.9896 - val_loss: 0.3120
- val_accuracy: 0.9764
```

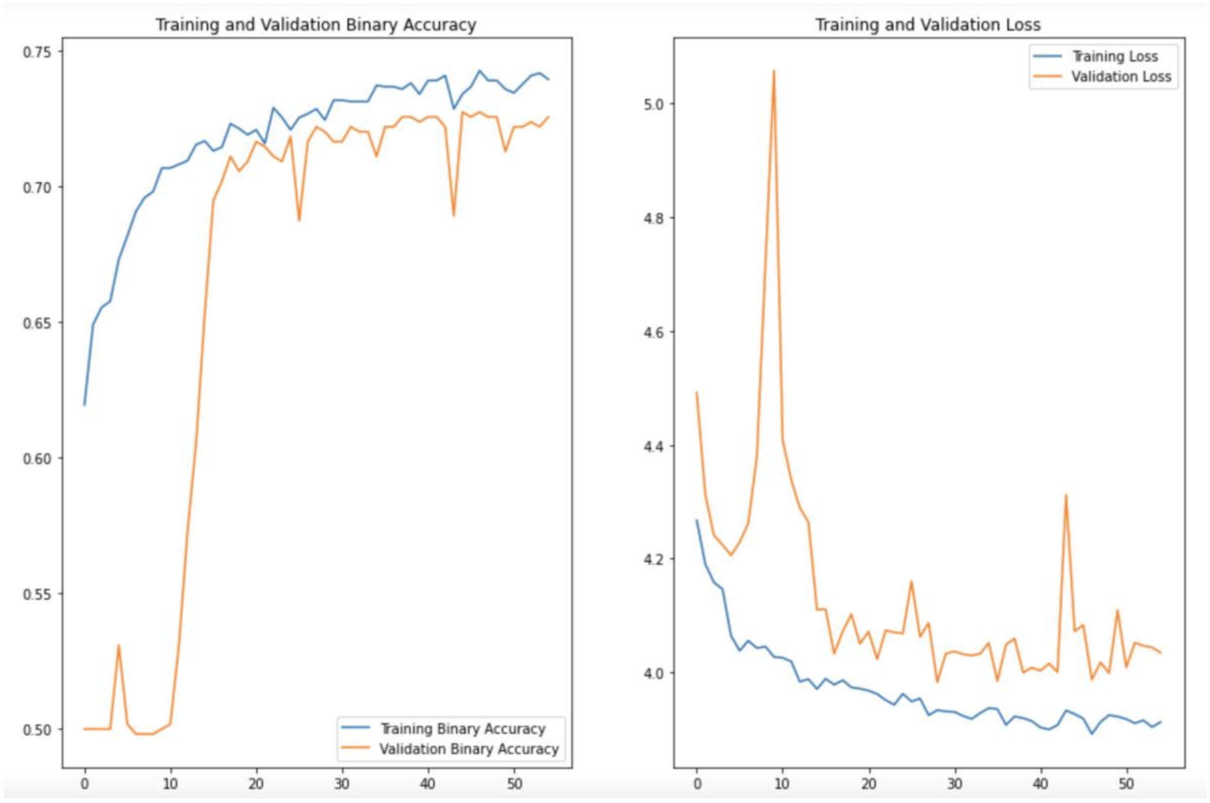
Each epochs time:
#1 around 30s
#2 around 20s
#3 around 30s

The graphs:

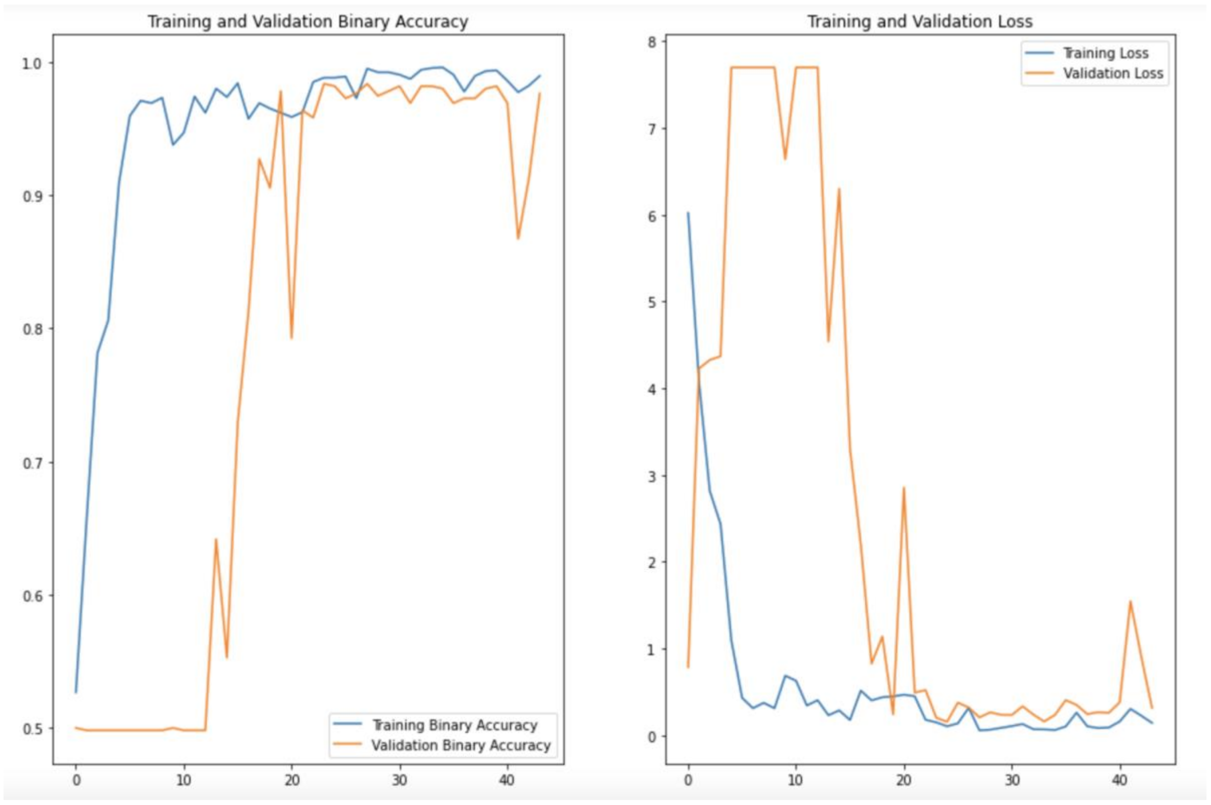
Model 1:



Model 2:



Model 3:



Therefore the Model #1 is good enough and the Model #3 slightly improved the performance.