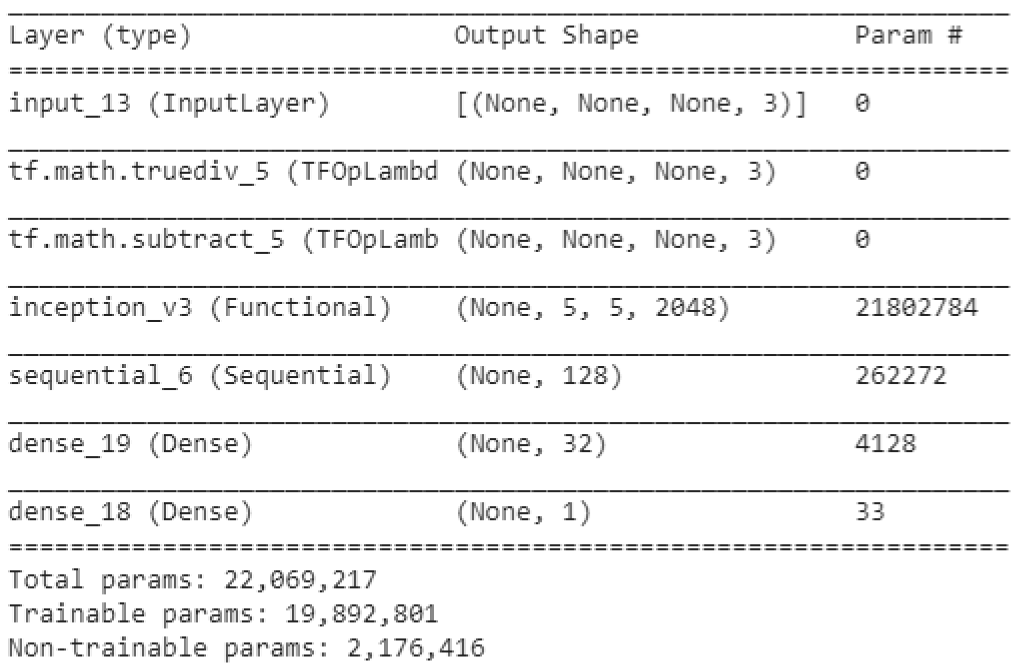
CS 542 Class Challenge: Image Classification of COVID-19 X-rays

Zhen Sha: U24418416
04/25/2021

Task1

1. **Part1: Architecture, optimizer, loss function, parameters**
   The input X-ray images are png files of size 224x224 with 3 channels. I use the rescaling and data-augmentation parameters provided in the template to ImageDataGenerator and flow them in batches of 25 using train_datagen.

   My base model is InceptionV3. I choose this as my base model because of its better performance on the ImageNet dataset than VGG16 and it has relatively fewer parameters compared with that. There are 311 layers in the pre-trained model, I use all the layers except for the last layer which is specific to the ImageNet problem, and I freeze 100 of them to be not trainable. The output layer dimension of InceptionV3 is (5,5,2048), and then I performed the GlobalAveragePooling2D layer to the output layer of the pre-trained model instead of Flatten because it gives me fewer parameters as a result of pooling. Then I add a Fully connected layer with 128 hidden units and Relu activation, a dropout layer with a 5% dropout rate, another fully connected layer with 32 hidden units and Relu activation, and a final sigmoid layer for classification. Here is a summary of the architecture:



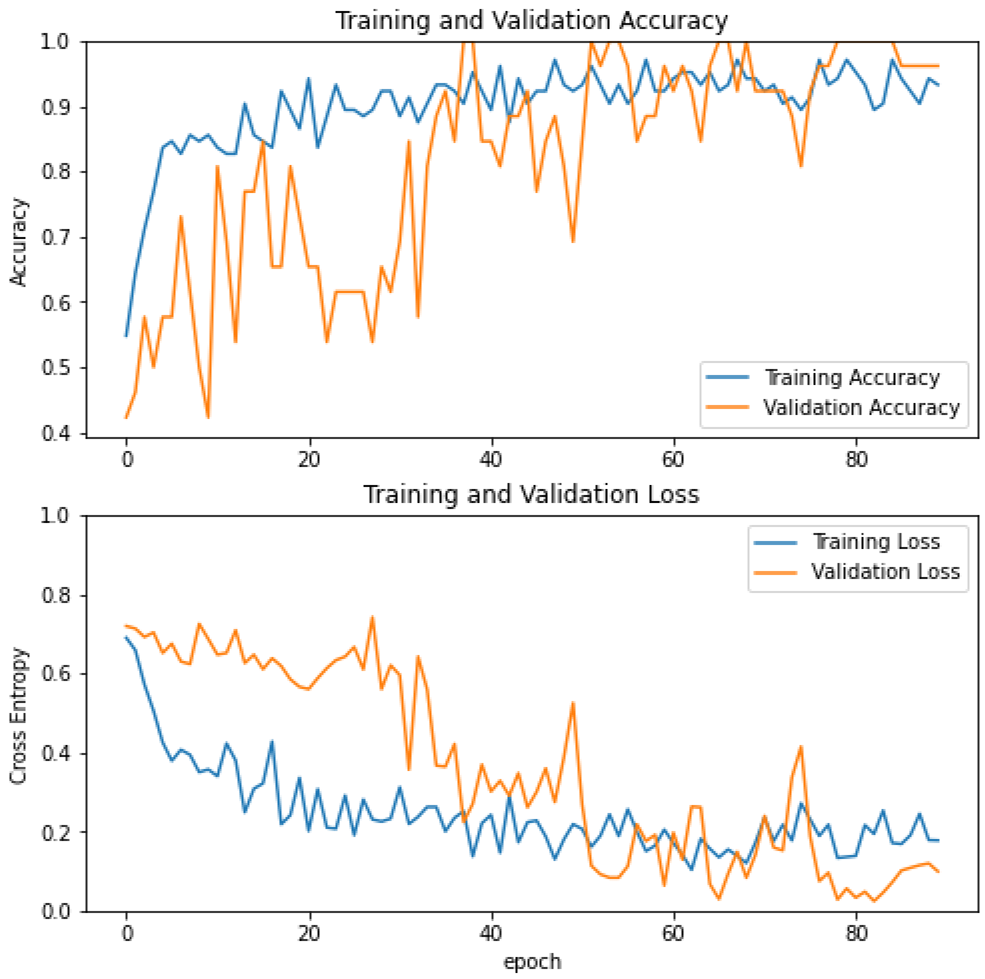
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        [(None, None, None, 3)]   0
_________________________________________________________________
tf.math.truediv_5 (TFOpLambd (None, None, None, 3)     0
_________________________________________________________________
tf.math.subtract_5 (TFOpLamb (None, None, None, 3)     0
_________________________________________________________________
inception_v3 (Functional)    (None, 5, 5, 2048)        21802784
_________________________________________________________________
sequential_6 (Sequential)    (None, 128)               262272
_________________________________________________________________
dense_19 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 33
=================================================================
Total params: 22,069,217
Trainable params: 19,892,801
Non-trainable params: 2,176,416
```

The dropout layer here works as regularization and I choose a 5% dropout rate after several tries. This means the dropout layer will randomly zero out 5% units each time. I found my model is unlikely to overfit in this task, this may have a relationship with my base model choice.

My optimizer is Adam with a learning rate of 0.0001. I started with a large learning rate of 0.001 and gradually decreased to 0.0001. I use the binary cross-entropy loss as my loss function since I am training a binary classifier. My error metric for this task is accuracy because this is a classification problem.

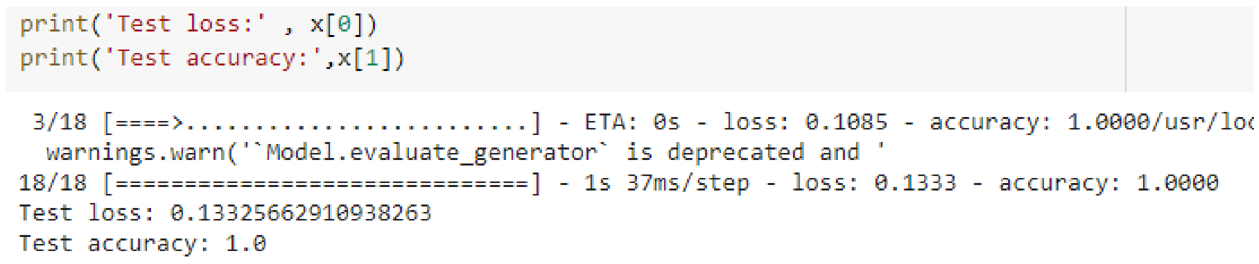
I have a total of 22,069,217 parameters in my model, with 19,892,801 of them trainable and 2,176,416 of them non-trainable.

2. **Part2: Accuracy and loss plots, test accuracy**
   I first use 40 epochs to train my model, but it is not enough for my model to converge, so I extend it to 90 epochs. I have attached the plots of accuracy and loss below.



In general, the trend of my plots is that the accuracy of training and validation datasets increases while the loss of them decreases. At epoch 36, the accuracy of training and validation sets both reaches 90% and above. From epoch 40 to the end, the training accuracy does not have obvious improvement, but the validation accuracy further increases and reaches 100% accuracy, and then it keeps relatively steady, not fluctuating too much from epoch 50 to the end. I find a corresponding trend on the loss plot. The training loss reaches the lowest range at about epoch 40, and the validation loss keeps decreasing until epoch 80. From epoch 50, the validation loss is lower than the training loss. I do not find the signal of overfitting in these plots. The overall trends of both training loss and validation loss are decreased.

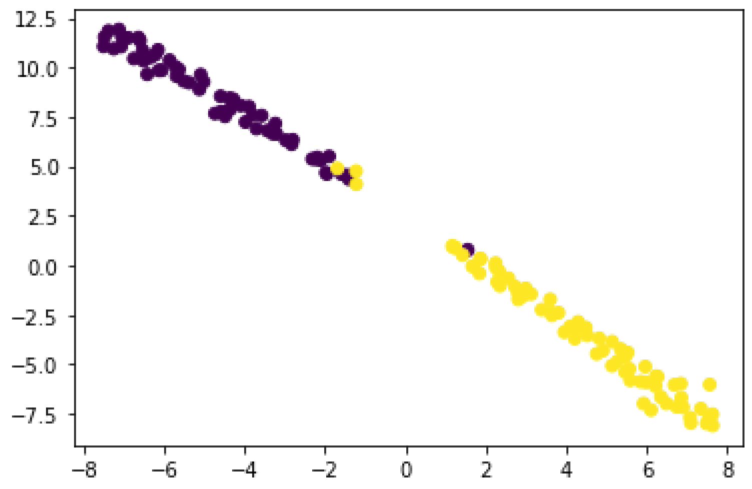
The test accuracy for this binary classifier is 100%.



```
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
 3/18 [====>.........................] - ETA: 0s - loss: 0.1085 - accuracy: 1.0000/usr/loc
  warnings.warn('`Model.evaluate_generator` is deprecated and '
18/18 [==============================] - 1s 37ms/step - loss: 0.1333 - accuracy: 1.0000
Test loss: 0.13325662910938263
Test accuracy: 1.0
```

3. **Part3: t-SNE visualization**
   I extract the features from the final layer before the output layer, and I use t-SNE to reduce the dimensionality of the extracted features to 2 dimensions. The resulting t-SNE visualization is attached below.



   The yellow points represent data in the Normal class and the purple points represent data in the Covid class. I can clearly see two clusters of data points, this is what I expect to see because there are two classes in my output. Although there are few data points being classified to the incorrect class because the normal X-ray images and the Covid-19 X-ray images share many similarities, overall the two clusters are separated well. This suggests that my extracted features are good and my binary classifier is trained well for this task.

Task2

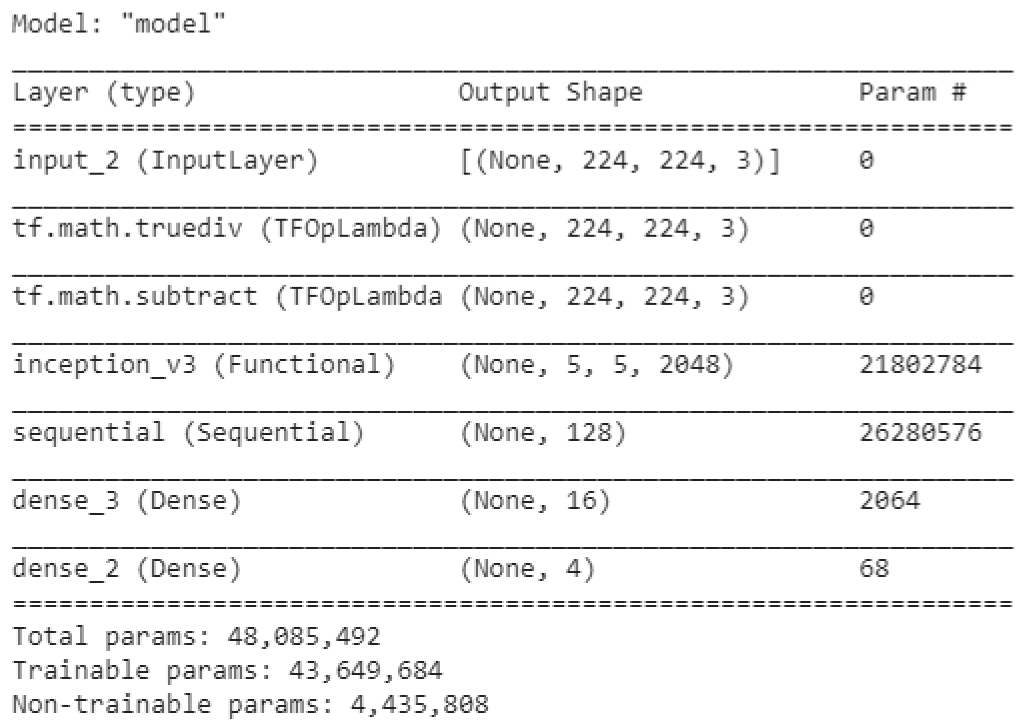
1. **Part1: Architecture, optimizer, loss function, parameters**
   The input X-ray images in this task are also png files of size 224x224 with 3 channels. I use the rescaling and data-augmentation parameters provided in the

template to ImageDataGenerator and flow them in batches of 20 using train_datagen.

Since VGG16 is a simple stack of convolutional and max-pooling layers followed by one another and a fully connected layer at the end but this task is more complicated than task1 (4-class vs. 2-class), I do not expect VGG16 to perform well in this case and I choose models (InceptionV3 and MobileNet) which can learn more complex features as my base models and compare which one is better.

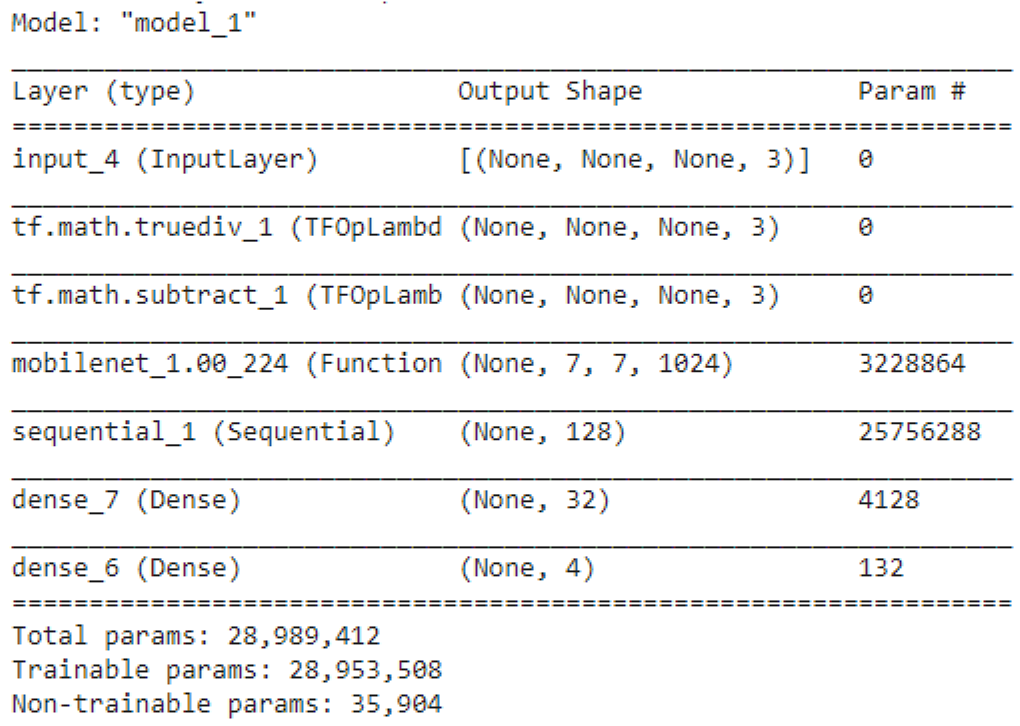
There are 311 layers in the pre-trained model InceptionV3, I freeze 150 of them after several tries while there are 86 layers in the MobileNet model and I freeze 20. To do the comparison, I use the same customized layers and hyperparameters such as the learning rate, batch size, etc. to train these models. The output layer shape of InceptionV3 is (5, 5, 2048) and that of the MobileNet is (7, 7, 1024). For the customized layers, I first add a Flatten layer to flatten the output layers from the pre-trained models to 1 dimension, then a fully connected layer with 512 hidden units and Relu activation, a dropout layer with a 50% dropout rate, a fully connected layer with 128 hidden units and Relu activation, a fully connected layer with 32 hidden units and Relu activation, and a final Softmax layer for multi-class classification. The Dropout layer works as regularization, I have tried several dropout rates and decided to use 50% because it performs better in terms of decreasing the probability of overfitting. My optimizer is Adam with a learning rate of 0.00001. Again, I started by using a large learning rate te 0.001 and gradually decreases to 0.00001. I use the categorical cross-entropy loss as my loss function instead of binary since my output has four classes this time. My error metric for this task is accuracy because this is still a classification problem.

There are 21,802,784 parameters in the output layer of InceptionV3 and 3,228,864 in MobileNet. For the model with InceptionV3 as the base, I have a total of 48,085,492 parameters with 43,649,684 of them trainable and 4,435,808 of them non-trainable. For the model with MobileNet as the base, I have a total of 28,989,412 parameters in the final model with 28,953,508 of them trainable and 4125 of them non-trainable. For both the base model and the final model, InceptionV3 has a much greater number of parameters than MobileNet.



```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0         
_________________________________________________________________
tf.math.truediv (TFOpLambda) (None, 224, 224, 3)       0         
_________________________________________________________________
tf.math.subtract (TFOpLambda (None, 224, 224, 3)       0         
_________________________________________________________________
inception_v3 (Functional)    (None, 5, 5, 2048)        21802784  
_________________________________________________________________
sequential (Sequential)      (None, 128)               26280576  
_________________________________________________________________
dense_3 (Dense)              (None, 16)                2064      
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 68        
=================================================================
Total params: 48,085,492
Trainable params: 43,649,684
Non-trainable params: 4,435,808
```

Model Summary: Model with InceptionV3



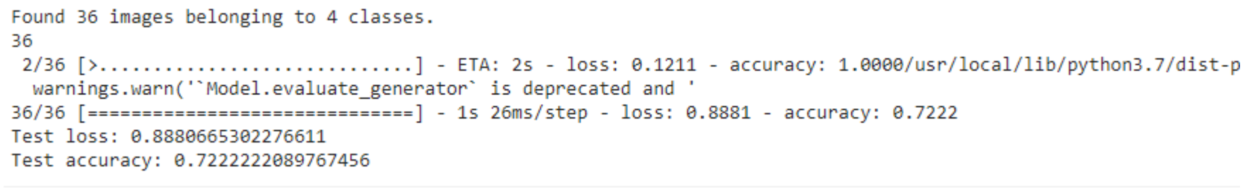
```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_4 (InputLayer)         [(None, None, None, 3)]   0         
_________________________________________________________________
tf.math.truediv_1 (TFOpLambd (None, None, None, 3)     0         
_________________________________________________________________
tf.math.subtract_1 (TFOpLamb (None, None, None, 3)     0         
_________________________________________________________________
mobilenet_1.00_224 (Function (None, 7, 7, 1024)        3228864   
_________________________________________________________________
sequential_1 (Sequential)    (None, 128)               25756288  
_________________________________________________________________
dense_7 (Dense)              (None, 32)                4128      
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 132       
=================================================================
Total params: 28,989,412
Trainable params: 28,953,508
Non-trainable params: 35,904
```

Model Summary: Model with MobileNet

2. Part2: Performance and best test accuracy

The test accuracy of the model with base model InceptionV3 is 72.22% while the one with base model MobileNet is 58%. In terms of architecture, the major difference between InceptionV3 and MobileNet is that MobileNet uses Depthwise separable convolution while InceptionV3 uses standard convolution. This difference results in a lesser number of parameters in MobileNet compared to InceptionV3, as shown in the previous section, MobileNet has 18,573,920 fewer parameters than InceptionV3 as a base model. As a result, MobileNet is faster to train, I end up using 220 epochs while I need 350 epochs for InceptionV3 while keeping the customized layers and hyperparameters (except the number of frozen layers) the same. However, The difference in their number of parameters could be one reason for MobileNethis to have worse performance.

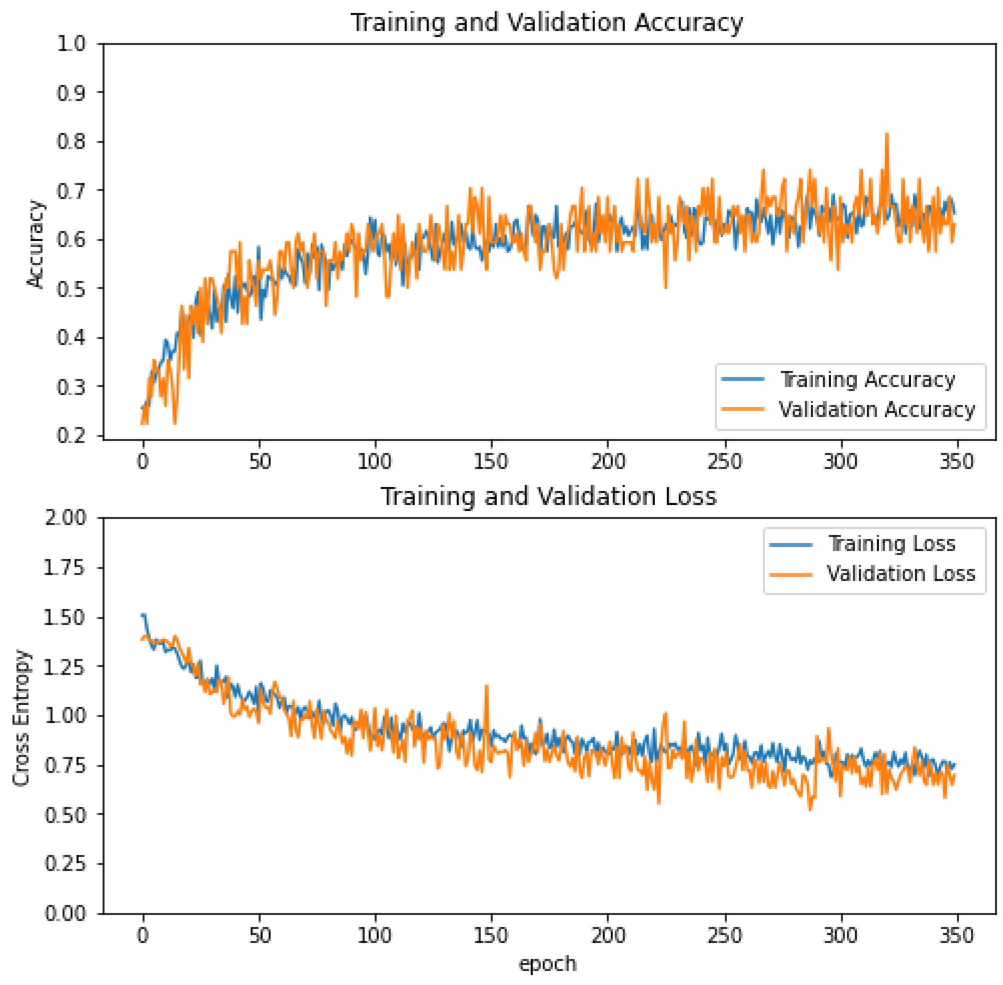
My best test accuracy is 72.22%.



```
Found 36 images belonging to 4 classes.
36
 2/36 [>.............................] - ETA: 2s - loss: 0.1211 - accuracy: 1.0000/usr/local/lib/python3.7/dist-p
  warnings.warn('`Model.evaluate_generator` is deprecated and '
36/36 [==============================] - 1s 26ms/step - loss: 0.8881 - accuracy: 0.7222
Test loss: 0.8880665302276611
Test accuracy: 0.7222222089767456
```

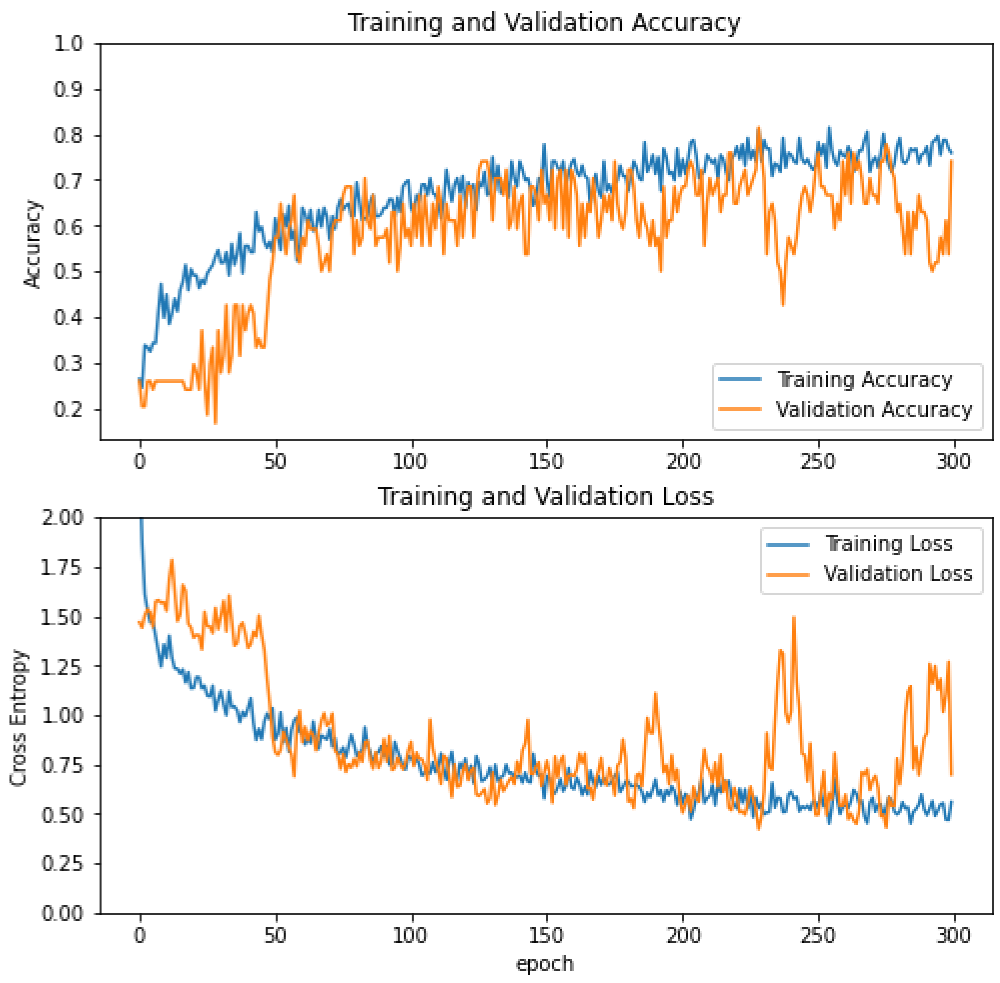
3. **Part3: Test and Loss plots**
   InceptionV3:
   I use a small learning rate of 0.00001, it works great on reducing the chance of overfitting but takes more epochs to train the model. From the plots, I can see that the validation loss is slightly lower than the training loss from the early beginning to the end while they have similar accuracy. The range that validation accuracy fluctuates is slightly greater than the training accuracy, the reason could be that we have a smaller number of data in the validation set and this makes it less stable. Overall, both training and validation accuracy increase slowly but steadily from epoch 0 to 250, and there is no significant improvement after epoch 250. Meanwhile, the loss keeps decreasing a little bit after epoch 250. I do not see the signal of overfitting here.

   I expect the MobileNet to learn in a smaller number of epochs, but I want to use epoch 300 as a starting point.



MobileNet
As I expected, MobileNet learns faster in terms of the training and validation accuracy, it uses about 50 epochs to reach an accuracy that over 60% while InceptionV3 takes about 100 epochs. However, the training and validation accuracy is not as synchronized as that in the InceptionV3 case. The validation accuracy is lower than the training accuracy with a large fluctuation. The training and validation losses also decrease much faster than the model with InceptionV3, they reach a loss of 0.75 and lower at around the 100th epoch. However, even the training loss decreases steadily, the validation loss fluctuates around it. At epoch 250, it seems like the model finds another local minimum and same for the epoch 350. But there is no further decrease in the loss in general.

Overall, the model with InceptionV3 has more stable performance.

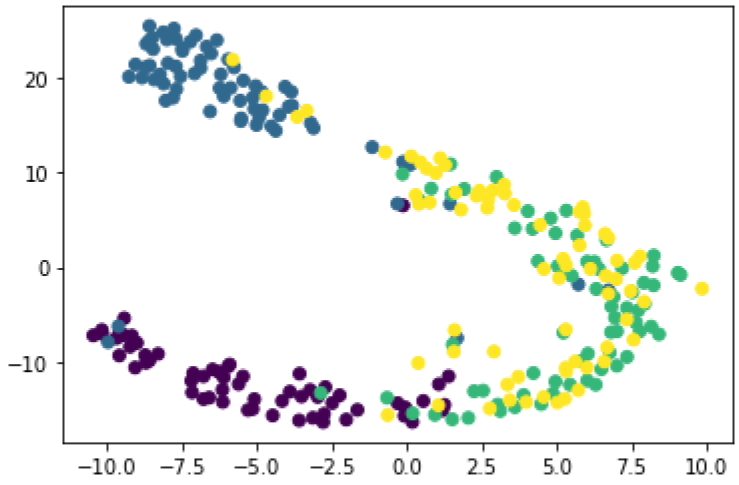


4.Part4: t-SNE

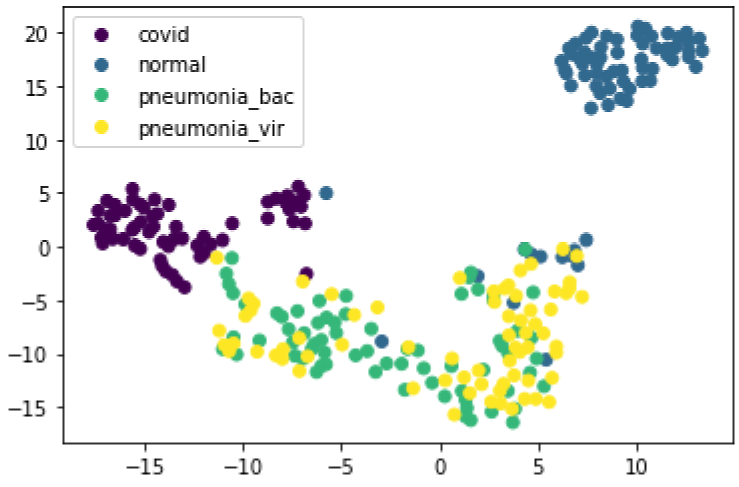
For both models, I extract the features from the final layer before the output layer, and I use t-SNE to reduce the dimensionality of the extracted features to 2 dimensions. The blue dots represent normal X-ray, purple represents the Covid-19 X-ray, green is the pneumonia_bac and yellow represents the pneumonia_vir.

The first plot is of the model with InceptionV3 and the second plot is of the model with MobileNet. From both plots, I can see four classes where two of them(pneumonia_bac and pneumonia_vir) cannot be separated well and another two (covid and normal) are easier to be separated. As t-SNE naturally expands dense clusters and shrinks spares ones, I cannot comment on the size of clusters. The distance between clusters may also mean nothing. I would say they both extracted good features in terms of separating the covid and normal classes.

Besides the overlaps between pneumonia_bac and pneumonia_vir, I also find that there are some overlaps between the normal and pneumonia_vir data, and some overlaps between covid and pneumonia_bac data, this may suggest that they share some similarities respectively. In terms of separating pneumonia_bac and pneumonia_vir, I can further try training a binary classifier to solve this problem.



t-SNE visualization: Model with InceptionV3



t-SNE visualization: Model with MobileNet