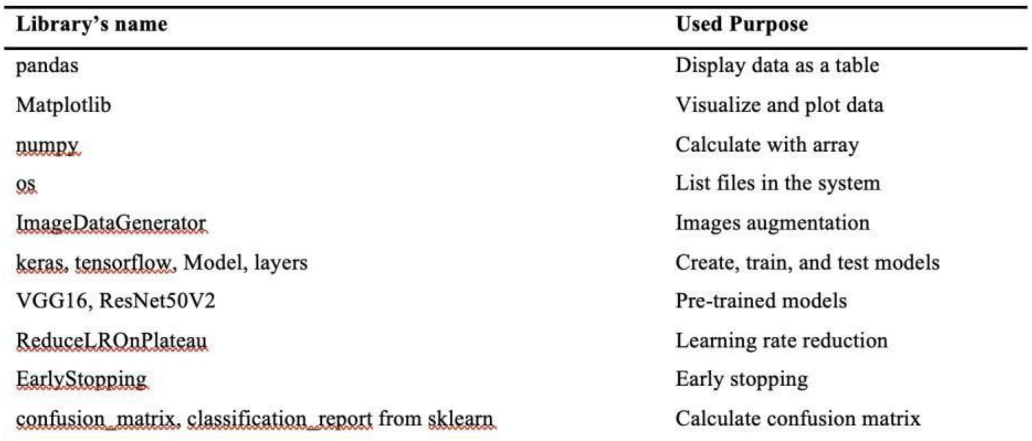
APPLIED AI MODULE COMP534 ASSIGNMENT 3

Student's name	Peerawit Mongkolchaidit	Saeth Wannasuphoprasit
ID	201556249	201585689

Note that 'number followed by topic name' is the section in the given python file. For example, '1.2 visualize train and test images' is the section number 1.2 in the given python file.

I. Introduction

The libraries used



Library's name	Used Purpose
pandas	Display data as a table
Matplotlib	Visualize and plot data
numpy	Calculate with array
os	List files in the system
ImageDataGenerator	Images augmentation
keras, tensorflow, Model, layers	Create, train, and test models
VGG16, ResNet50V2	Pre-trained models
ReduceLROnPlateau	Learning rate reduction
EarlyStopping	Early stopping
confusion_matrix, classification_report from sklearn	Calculate confusion matrix

The detail of the development process

We are given image datasets consist of train and test dataset for 3 different types of images, Pneumonia, Covid-19, and Normal stage. We are required to choose 2 CNN pre-trained models out of 5 to analyze these images and see which CNN pre-trained models bring out the best accuracy. First, '1.1 download dataset', we need to install Kaggle API to download image datasets and make directory named Kaggle to implement on this project and we need to unzip those files. After that we used imageplot function to distinguish 3 types of images as shown in '1.2 visualize train and test images. The critical process in which we do after image plotting is Image Augmentation '1.3 images augmentation', this method allows our CNN pre-trained models to learn various possible angles of these images and still be able to classify which image is which. All functions used in Images Augmentation process are listed below with explanations.

1. samplewise_center=True: to set images on the center.
2. samplewise_std_normalization=True: to normalize images pixel.
3. width_shift_range=0.2: random-shifted images either towards left or right.
4. height_shift_range=0.2: just as width_shift_range, but images shifted vertically instead.
5. shear_range=45: slant or tilt the shape of the images, differently from rotation. In shear function we must fix one axis and stretch images at a particular angle.
6. zoom_range=0.2: model captures images from various zooming dimensions.
7. rotation_range=20: images randomly rotated by the angles of set range.

Later we also need to generate train dataset, we have set batch size to 10, the reason why we set batch size to 10 is because if we set it too low, it would give our CNN pre-trained models more time to train but, in another way, if we set it too high, system memory won't be sufficient, so we figured 10 is the perfect number for batch size. Opposite way when we generate our test dataset, we only set batch size to 1 since we can only test 1 image at a time. These steps result in total of 5144 images for train dataset and 1288 images for test dataset, these two train/test datasets combined to 3 classes labeled as (Normal: 0, Pneumonia: 1, Covid-19: 2). We set target image size to be (300, 300) to reduce complexity of the original images and it is suitable to fit in the pre-trained models in our work according to the guide line in VGG16, and ResNet50V2.

We have chosen VGGNet and ResNet CNN pre-train models to train our image datasets, the reasons behind our decision are, firstly according to research paper, it dealt with wrists x-ray images, which we believe they are relatively in the similar category as given image datasets from our assignment. We have calculated the mean of all 5 models and it result to us that VGGNet and ResNet gave the best accuracy in comparison to other 3 CNN pre-trained models, secondly according to the instructions in this assignments we are required to choose the least complex CNN pre-trained models, so why specifically we chose “VGG16 and ResNet50V2” is that these 2 CNN pre-trained models from Keras information are overall the least complex CNN pre-trained models, from our research VGG16 and VGG19 in comparison, VGG16 has 138.4M parameters while VGG19 has 143.7M parameters. Regarding ResNet models, although ResNet50 and ResNet50V2 have the exact same number of hyperparameters, but in terms of Depth (16 to 19), Time (ms) per inference step (CPU: 45.6 to 58.2, GPU: 4.4 to 4.6), ResNet50V2 has less numbers on those areas which makes ResNet50V2 faster when it comes to train models. Next process, we would like to find out which of these 2 CNN pre-trained models perform better, so we compared these 2 models by Transfer Learning method, and we created the models by

1. Eliminate top-layers of these 2 CNN pre-trained models.
2. Freeze the weights of these 2 CNN pre-trained models to default value.
3. Add layers on top of our CNN pre-trained models which is called Flatten Layer in order to transform from 2D Convolutional layers to Fully Connected layer (in this case is 1 Dense layer with 3 nodes output to match our classification task that has 3 classes).

• The function that we use to implement the models is stated in ‘2.1 train and evaluation functions’ and the name of the function is “pretrained_model”.

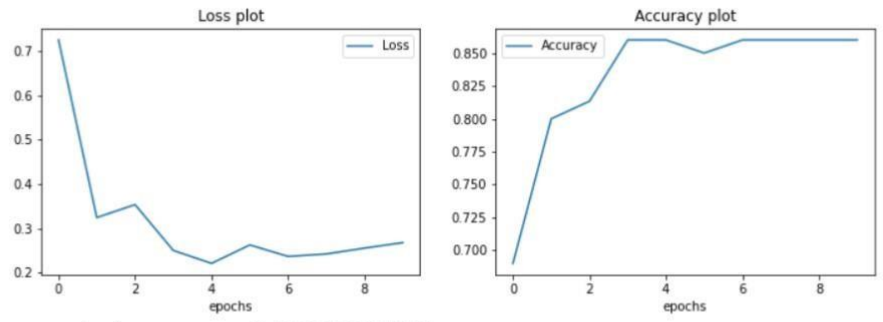
Later we used these methods to train our CNN pre-trained models with these configurations:

1. Utilized Adam Optimization according to lecture slide week 6, because it gave faster training time (less epochs).
2. Chose loss function to binary_crossentropy, since it is suitable for computing loss between true labels and predicted labels as our reference.
3. Utilized matrices to accuracy because we would like to track accuracy rate of the models.
4. Set epochs and step per epoch to 10 and 30, because it is sufficient to train our models (both models achieve more than 80% test accuracy with this configuration) and we needed not to waste time training the models (high epoch and step per epoch will unnecessarily give too much time to train our models).

II. Results description

Evaluation Results with detailed explanation

VGG16

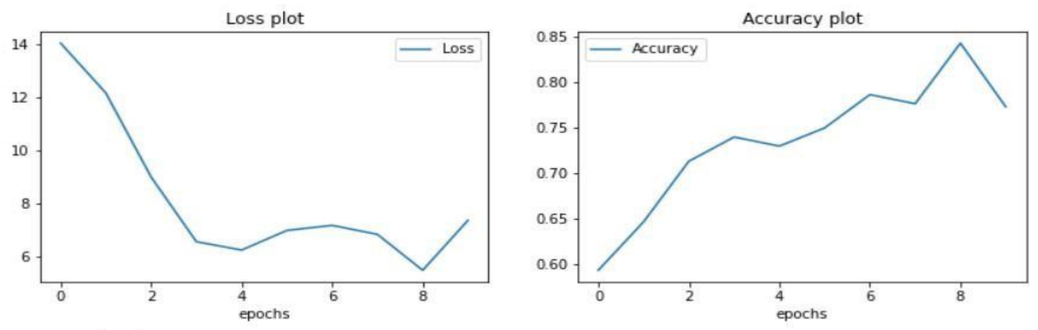


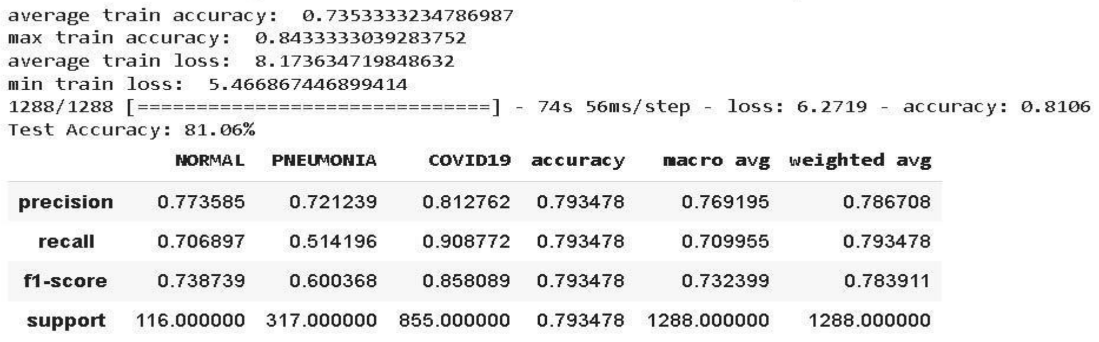
```
average train accuracy:  0.8313333451747894
max train accuracy:  0.8600000143051147
average train loss:  0.3136019244790077
min train loss:  0.22081007063388824
1288/1288 [==============================] - 83s 64ms/step - loss: 0.3261 - accuracy: 0.8323
Test Accuracy: 83.23%
```

	NORMAL	PNEUMONIA	COVID19	accuracy	macro avg	weighted avg
precision	0.788732	0.635983	0.983533	0.833075	0.802750	0.880451
recall	0.965517	0.958991	0.768421	0.833075	0.897643	0.833075
f1-score	0.868217	0.764780	0.862771	0.833075	0.831923	0.839144
support	116.000000	317.000000	855.000000	0.833075	1288.000000	1288.000000

Figure 1: Training loss (top left), training accuracy (top right), test accuracy (middle), confusion matrix (bottom) available in ‘2.2 VGG16 ’.

ResNet50V2





```
average train accuracy:  0.7353333234786987
max train accuracy:  0.8433333039283752
average train loss:  8.173634719848632
min train loss:  5.466867446899414
1288/1288 [==============================] - 74s 56ms/step - loss: 6.2719 - accuracy: 0.8106
Test Accuracy: 81.06%
```

	NORMAL	PNEUMONIA	COVID19	accuracy	macro avg	weighted avg
precision	0.773585	0.721239	0.812762	0.793478	0.769195	0.786708
recall	0.706897	0.514196	0.908772	0.793478	0.709955	0.793478
f1-score	0.738739	0.600368	0.858089	0.793478	0.732399	0.783911
support	116.000000	317.000000	855.000000	0.793478	1288.000000	1288.000000

Figure 2: Training loss (top left), training accuracy (top right), test accuracy (middle), confusion matrix (bottom) available in '2.3 ResNet50V2'

REMARK of our main purpose of this project

In this case, we want high recall score in class Pneumonia and class Covid-19, because we need low false negative percentage (predicted negative for Covid/ Pneumonia but actually positive) but in other way, we want precision score on class Normal to be high, because we want low false positive percentage (predicted that a person is normal but actually not). Furthermore, in train accuracy, test accuracy and f1-score we should choose the highest value possible, because it is the value that indicates accuracy on prediction that which models predict the most accurate.

For our CNN pre-trained models, in **Figure 1 and Figure 2**, we decided to choose VGG16 since it performs well in every factor, such as train accuracy (85% vs 80%), test accuracy (83.2% vs 81.1%), macro average f1-score (83.2% to 73.2%). For precision and recall, VGG16 outperforms ResNet50V2 in every class (only recall in ResNet50V2 in class Covid-19 that performs better than VGG16 (90.9% vs 76.8%)). In comparison the re-call score for class Pneumonia and class Covid-19 are higher (95.8 to 51.4) and (76.8 to 90.8) while precision score for Normal class (78.8 to 77.3).

Given the result, we have chosen VGG16 to be our improved model, since only the recall rate is lower on Covid-19 class, apart from that all the scores are higher.

VGG16 is overfitted according to accuracy graph in **Figure 1**, train accuracy is roughly around 85%, test accuracy around 83%. This dues to the fact that epochs that was used for training is too high, giving the model captures too much feature that it wouldn't generalize well in the test dataset. Thus, in the accuracy graph, from epoch = 4, it is going up towards top accuracy already, so we have no need to train any further compared to ResNet50V2 in **Figure 2** that needed to be trained until epoch 9 to 10 then it starts leading up to top accuracy and it still is not overfitted, because from the accuracy graph in **Figure 2**, train accuracy is roughly around 70% to 80% and test accuracy is around 81.1%.

Improve the previous VGG16 (in '3. Improve the previous VGG16')

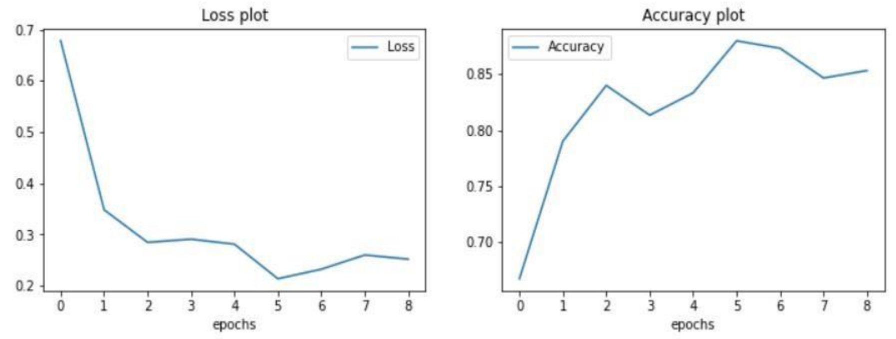
Next, we improved VGG16 by bringing VGG16 base model (freeze default weights) and add more layers as stated below.

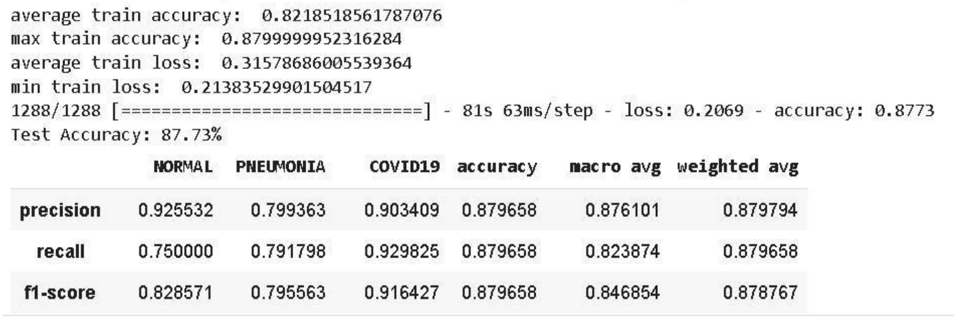
1. Added Flatten layer, in order to transform from 2D Convolutional layer to 1D Fully Connected layer.
2. Added Batch Normalization layer to normalize weights from output layer of VGG16 base model, to decrease bias that might occur, for example there might be some weights that are higher than the other, making more impact on prediction to some features more than other features and might lead to model misprediction that causes the model's performance to drop. Apart from this, batch normalization makes convergence faster, since it would take fewer epochs to reach the same performance (according to lecture slide week 5). In this layer, we must freeze the weights when training the model, in order to be able to use feature of batch normalization correctly by this tensorflow tutorial.
3. Add dropout layer to reduce overfitting (as shown in **Figure 1** accuracy graph), it reduces the number of active neurons during the training process, meaning that it helps reducing models' complexity and that leads to better model generalization for test dataset. The reason we set dropout ratio to 0.3 was that if we set higher, the amount of active neurons that were blocked will be too high and will make the model underfitted (model doesn't capture feature good enough to classify images) in other way, if we set it lower the model will have too many active neurons and will make it overfitted (model captures too much information making it not generalize well in real world situation).

These are the steps of training model

1. Compile model in the same way as in '2.1 train and evaluation functions' entirely (optimizer, loss function, matrices).
2. Utilize learning rate reduction in every 1 epoch when accuracy doesn't improve. Learning rate in the beginning can obtain high values for the most efficient model learning, but when keeps on training, model starting to have steady accuracy, we should reduce learning rate to focus more in finding the best parameters to get the more accurate accuracy.
3. Utilize Early Stopping when accuracy doesn't improve to stop training model when accuracy doesn't increase for 3 epochs consecutively. The reason we selected 3 epochs was because, in case the accuracy fluctuates up and down, so we set the accuracy not to reduce nor improve for 3 times before it stops training, by doing so, we can save resources in training model and be able to reduce the occurrence of overfitting in '2.2 VGG16', because it is the process of limiting the model to overlearn, that won't let the model capture too much information and generalize well in test dataset.
4. Train model by setting epochs and step per epoch to 10 and 30 as implemented in '2.2 VGG16'.

These are the results of train added layers (3.1 train added layers)





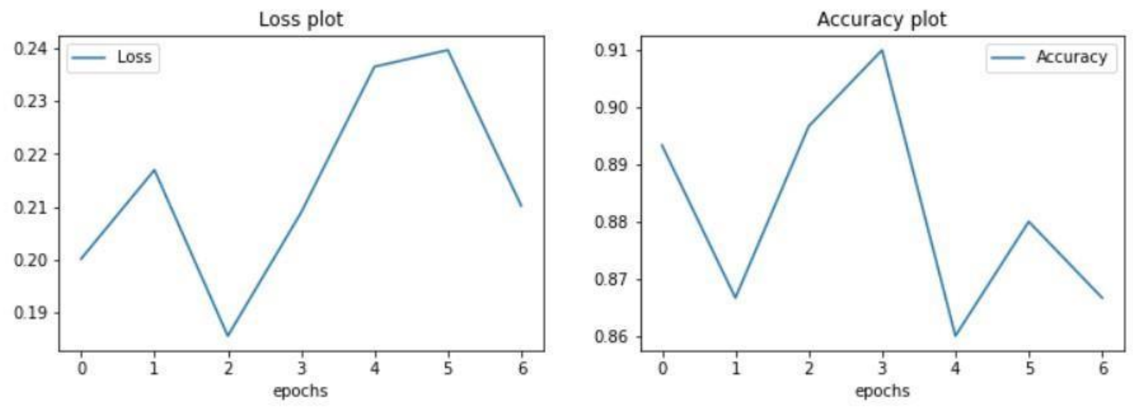
```
average train accuracy:  0.8218518561787076
max train accuracy:  0.8799999952316284
average train loss:  0.31578686005539364
min train loss:  0.21383529901504517
1288/1288 [==============================] - 81s 63ms/step - loss: 0.2069 - accuracy: 0.8773
Test Accuracy: 87.73%
```

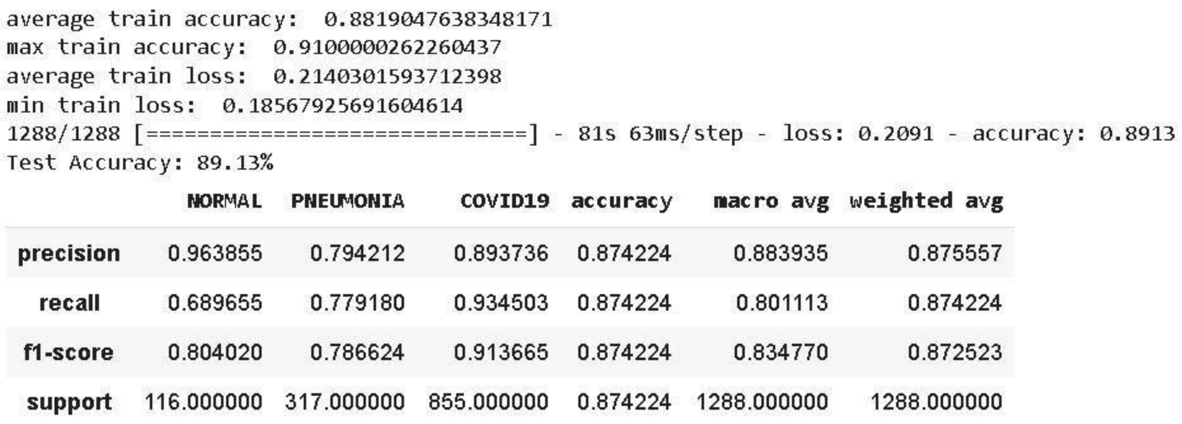
	NORMAL	PNEUMONIA	COVID19	accuracy	macro avg	weighted avg
precision	0.925532	0.799363	0.903409	0.879658	0.876101	0.879794
recall	0.750000	0.791798	0.929825	0.879658	0.823874	0.879658
f1-score	0.828571	0.795563	0.916427	0.879658	0.846854	0.878767

Figure 3: Training loss (top left), training accuracy (top right), test accuracy (middle), confusion matrix (bottom) available in '3.1 train added layers'

From **Figure 3**, train accuracy is not much different than the previous one **('Figure 1'),** which is roughly around 85%, while test accuracy has improved from the previous (83.2% to 87.7%) making it not overfit anymore because of the methods Batch Normalization, Dropout, Learning Rate Reduction and Early Stopping that mentioned above. The result of Learning Rate Reduction and Early Stopping can be visualized in '3.1 train added layers', seeing it clearly that learning rate starts from 0.001 and the moment when model doesn't improve accuracy in epoch 4, learning rate has been reduced to 0.0001 and ongoing to $1*10^{-7}$in epoch 9 (stopped training since Early Stopping has noticed that the accuracy doesn't improve in 3 consecutive epochs). Apart from that, in class Normal the model improved precision from 78.9% to 92.6% that basically answers our criteria which we wanted in *** REMARK of our main our purpose of this project section *** mentioned above. In class Pneumonia, recall score reduced from 95.9% to 79.2%, meaning that it is not quite a decent improvement, because in this class we need high recall score, in the opposite way, recall score from class Covid-19 increases from 76.8% to 93.0% which means it was a relatively decent improvement.

Next process, fine-tuning parameters in layers in the base model (2D Convolutional layers 15th-17th layer) to improve model accuracy '3.2 fine-tune the layers inside VGG16'. We freeze weights that are left, including weights from added layers that have been trained before in '3.1 train added layers 'at early point to keep important features and continue to train the model from the latest epoch that has been trained before. The result is shown below.





```
average train accuracy:  0.8819047638348171
max train accuracy:  0.9100000262260437
average train loss:  0.2140301593712398
min train loss:  0.18567925691604614
1288/1288 [==============================] - 81s 63ms/step - loss: 0.2091 - accuracy: 0.8913
Test Accuracy: 89.13%
```

	NORMAL	PNEUMONIA	COVID19	accuracy	macro avg	weighted avg
precision	0.963855	0.794212	0.893736	0.874224	0.883935	0.875557
recall	0.689655	0.779180	0.934503	0.874224	0.801113	0.874224
f1-score	0.804020	0.786624	0.913665	0.874224	0.834770	0.872523
support	116.000000	317.000000	855.000000	0.874224	1288.000000	1288.000000

Figure 4: Training loss (top left), training accuracy (top right), test accuracy (middle), confusion matrix (bottom) available in '3.2 fine-tune the layers inside VGG16'

Apparently, train accuracy has increased from 85% to 88% and test accuracy has increased from 87.7% to 89.1%, due to the process of fine-tuning layers within base model to make the model captures more important features, in order to improve the performance of the model while still utilizing Learning Rate Reduction and Early Stopping to reduce overfitting phenomenal as mentioned above. Noticeably from '3.2 fine-tune the layers inside VGG16', learning rate starts from
$1*10^{-8}$in epoch 8 and when the model doesn't improve accuracy in epoch 11, learning rate was set down to $1*10^{-9}$ongoing to $1*10^{-8}$ in epoch 15 (stopped training since Early Stopping notices that accuracy doesn't improve in 3 consecutive epochs). Moreover, in class Normal precision score has increased from 92.6% to 96.4% which is considered decent according to *** REMARK of our main our purpose of this project section *** mentioned above. In class Pneumonia, recall score has decreased slightly from 79.2% to 77.9% which doesn't seem so good while class Covid-19 has slightly increased from 93.0% to 93.5%.

Finally, we have concluded that VGG16 base model performed very decent in terms of train accuracy, test accuracy, f1-score, class Normal (precision), class Covid-19 (recall), but didn't perform quite well in class Pneumonia (recall). Results of Batch Normalization, Dropout, Learning Rate Reduction and Early Stopping could really help reducing overfitting phenomenal and help the performance of the model. Apart from that, fine-tuning parameters in layers of base model could really improve the performance of the base model.

Any future work necessary to improve the current model

1. We can furthermore train layers inside base model that hasn't been fine-tuned to make the model predict more accurately.
2. We might be able to put more added layers on top of base model, such as CNN, Maxpooling or even increase Dense layers just before output layer, in order to improve the model hopefully, but this can't be guaranteed.
3. We could also consider adding more batch normalization layers within layers in base model, to normalize weights in each layer and reduce weight's bias that might occur during training process to increase the performance of the model.

III. Final conclusions

The challenges of the project encountered

1. Cleaning images process was quite challenging, since we didn't know the most suitable modification/augmentation to our project, such as we didn't know the exact numbers to set for shear_range and rotation_range functions, so we had to try different numbers and see how it turns out as we wanted.
2. Time consuming on training deep learning models and really used up quite a lot of resources such as memory, CPU and GPU.
3. The process of choosing the right CNN pre-trained models or architectures to train in our project was quite difficult, so we managed to look for reference of various research papers mentioned above in this report.
4. To improve precision score and recall score was relatively challenging, especially for multi-class that in each class has matrices we wanted to have high score (precision, recall) differently. Of all the techniques and methods mentioned, accuracy, f1-score, precision (class Normal), recall (class Covid-19) the results were all decent but the recall score of class Pneumonia dropped, it doesn't guarantee that all the techniques and methods used in this project would really bring out the best precision and recall scores as we desired.

Task allocation

Task	Person in charge	Deadline for each task
Data Preparation	Saeth and Peerawit	02/05 – 04/05
Visualize train and test images	Saeth	02/05 – 02/05
Image Augmentation	Saeth and Peerawit	03/05 – 04/05
Pre-trained models	Saeth	04/05 – 05/05
Improvement of VGG16	Saeth	05/05 – 05/05
Report writing	Saeth and Peerawit	06/05 – 10/05