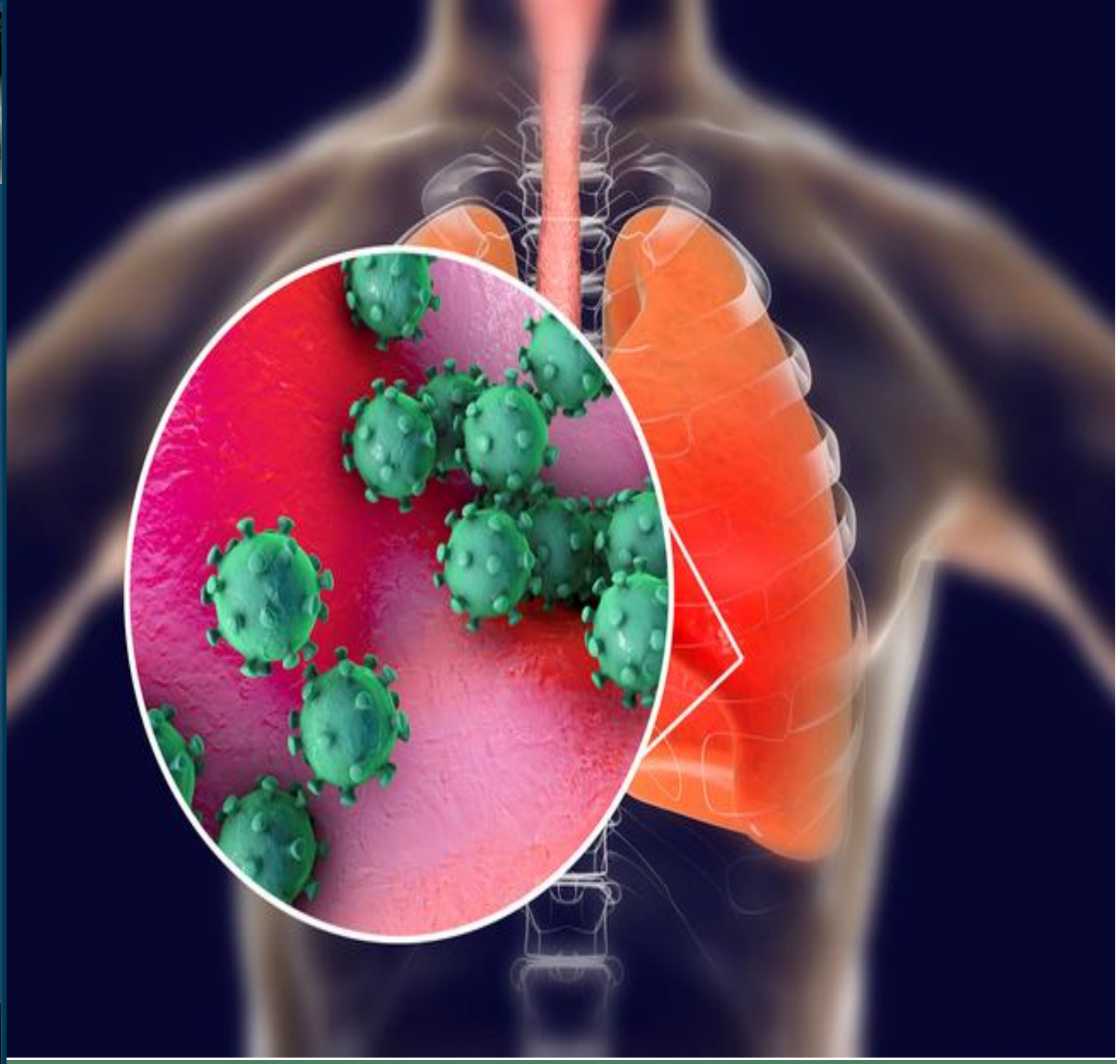


Use CNN to predict the likelihood of COVID-19 using chest X-ray images

Course work Case study Submitted to Amity Future Academy by

1. **Rajagopal Krishnamurthy**
2. **Sudi Abubakari**
3. **Aditya Kuche**

1. Introduction

COVID-19 is a life-threatening infectious disease caused by a newly discovered virus called noval corona virus (2019-nCOV). The disease erupted in late 2019 and spreads across countries across all continents affecting many people and has taken more lives. Due to high spread of disease, traditional diagnosing the disease like inspection of physical symptoms, clinical evidence consumes time.

Facts and Statistics

As on 6th March 2021 total number of cases reported across the globe is 116 million and the Mortality rate is 2.22%.

Source: https://www.worldometers.info/coronavirus/

X-Ray review is one of the preliminary diagnostic methods to identify a patient is infected with Covid-19. Manual review may cause variation in interpreting the results and will lead multiple rounds of test, at time patient may not have time to take those diagnosis. So, a precise, simple and automated solution warranted to tackle the current situation to support the preliminary screening for Covid-19.

It is time that it is required the urgent establishment of precise and fast diagnostic tests to verify suspected cases, screen patients, and conduct virus surveillance. This establishment of precise, fast and inexpensive diagnostic tests methods can lower the risk of spreading infection, alleviate the strain on the healthcare system, and mitigate the cost of care for both individuals and the government and improve the health care system in addressing the disease. In this study, a focus on using medical images to diagnose Covid 19 will be established where by an algorithm will trained to analyze the images and classify patients to either have Covid 19 or not.

2. Project Objective:

The Objective of this course work is to create a machine learning algorithm to automatically detect the Covid-19 infection using digital chest X-Ray.

3. Role of Machine Learning:

Machine learning plays an important role in detection of Corona virus because other methods are slow lead to the severe spread of the pandemic causing more death.
Following are the methodology / steps planned

1. Apply pre-trained Convolutional Neural Network to classify Normal and Covid-19 X-Ray

2. Use the collected X-rays to train the model and apply the transfer learning process to classify the type of chest X-rays.
3. Classification Accuracy will be used as key performance metrics to gauge the performance of CNN model.

If model comes with high accuracy will help to perform the preliminary screening for potential Covid-19 patients by the radiologist

4. Data Exploration

Large number of X-ray images are publicly available for machine learning and data research purpose in Kaggle. Kaggle with enormous volume of more than 50,000 public dataset across various areas for machine learning analysis has helped to have dataset for this study. The dataset has four categories of the images of the chest X-rays namely

- Covid-X-rays of COVID positive cases patients
- Viral Pneumonia- X-rays of patients shown pneumonia but not COVID
- 'normal'- They have neither presented the symptoms of COVID nor pneumonia
- Pulmonary opacification

All the images are in Portable Network Graphics (PNG) file format and resolution are 299*299 pixels

Source: https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

5. Feature Engineering

In this case study we have chosen VGG16. This model has won first prize in ILSVR competition in 2014. It is considered to be one of the excellent model architectures till date. It is a sequential model, so all the layers are arranged in a sequential order. From Keras ImageDataGenerator function is imported, since it has many image manipulation functions to rescale, rotate, zoom, flip etc. This function will not impact the data stored on the disk. The objective of ImageDataGenerator is to import data with labels easily into the model.

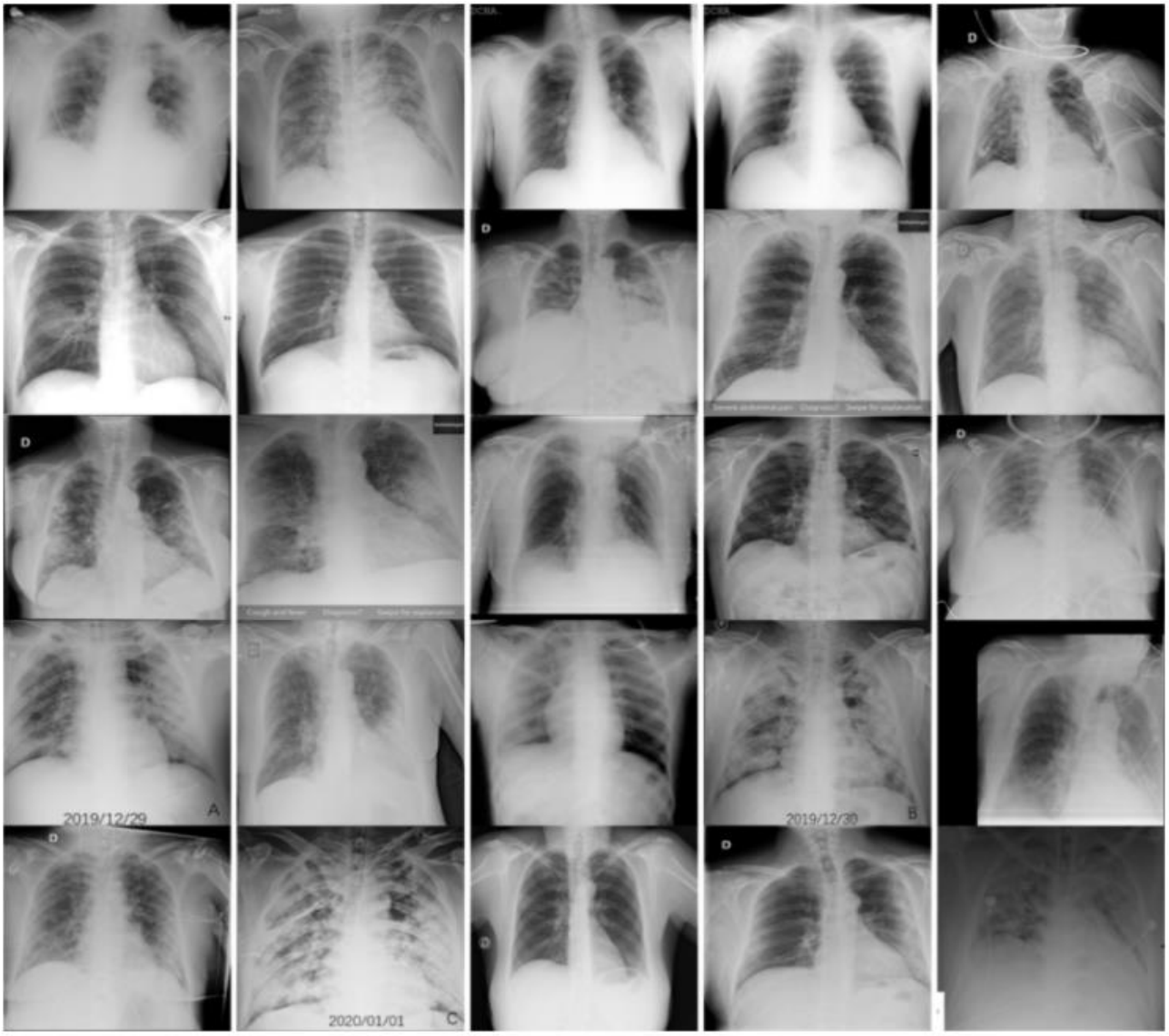
```
In [12]: vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

In [13]: for layer in vgg.layers:
             layer.trainable = False

In [14]: folders = glob('D:/DL/Covid-19/Covid-New/training_set/*')
         x = Flatten()(vgg.output)

In [15]: prediction = Dense(len(folders), activation='softmax')(x)
         # create a model object
         model = Model(inputs=vgg.input, outputs=prediction)
         # view the structure of the model
         model.summary()
```

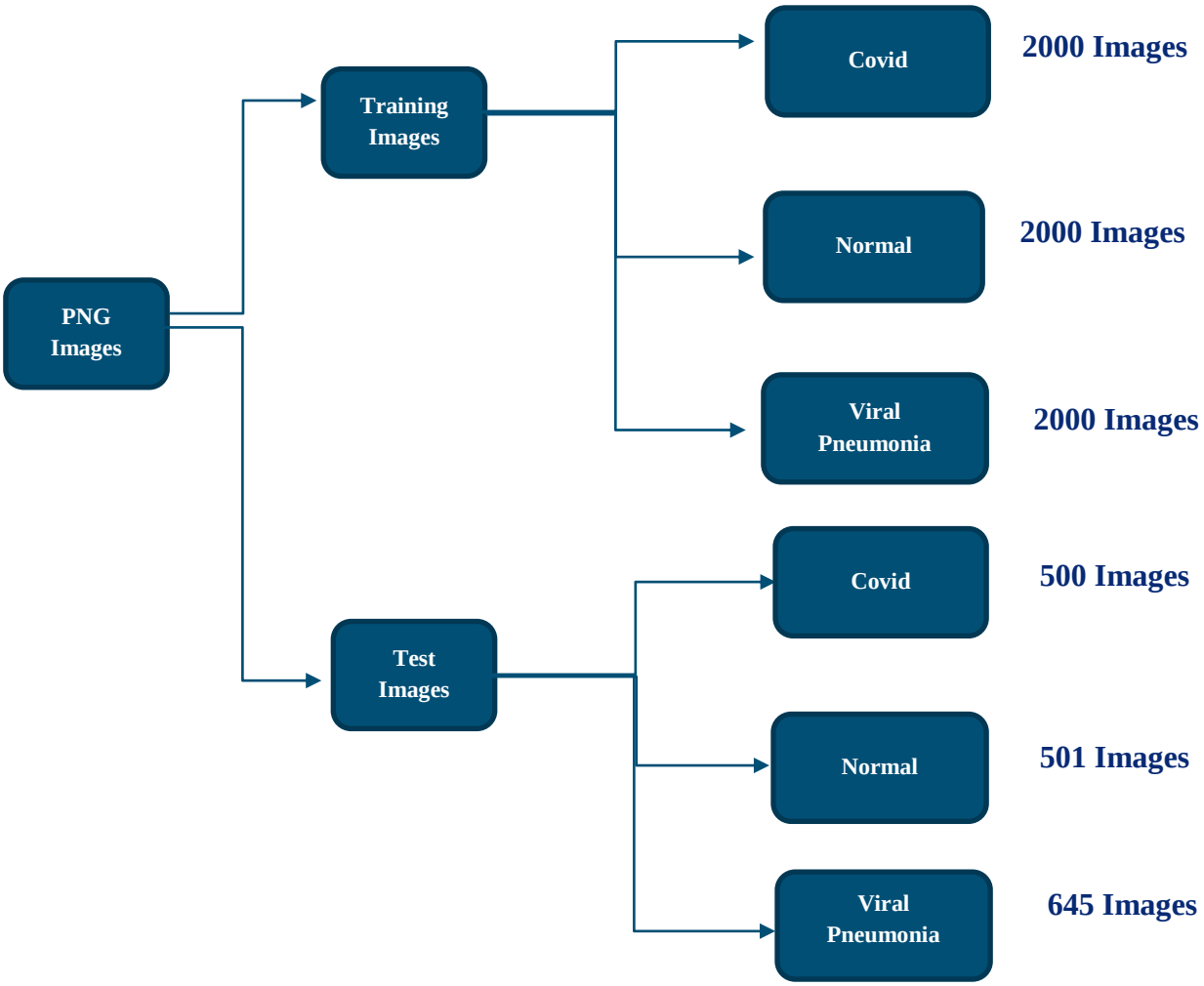
6. Building Training/Validation/Test Samples

Team went ahead and created folders for training and test data set. ImageDataGenerator is used to Object both training and test data sets. The folder structure of the data as follows. IDG has the ability to label all the data inside the categorized folders, so data set will be readily prepped for neural network processing.

```
IMAGE_SIZE = [224, 224]

train_path = 'D:/DL/Covid-19/Covid-New/training_set'
valid_path = 'D:/DL/Covid-19/Covid-New/test_set'
```

In [11]:
```
# Plot Multiple Images
bunchOfImages = imagePatches
i_ = 0
plt.rcParams['figure.figsize'] = (225.0, 225.0)
plt.subplots_adjust(wspace=0, hspace=0)
for l in bunchOfImages[:25]:
    im = cv2.imread(l)
    im = cv2.resize(im, (224, 224))
    plt.subplot(5, 5, i_+1) #.set_title(l)
    plt.imshow(cv2.cvtColor(im, cv2.COLOR_BGR2RGB)); plt.axis('off')
    i_ += 1
```





7. Model Selection Execution

After initialization, all the parameters and features are applied sequentially. After creating all the convolution, data passed to the dense layer. We have used 4 units of dense layer in the end with SoftMax activation as we have 2 classes to predict from at the end which are Covid-19 and Normal. The SoftMax layer will output the value between 0 and 1 based on the confidence of the model that which class the images belongs to.

In [12]:
```
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

In [13]:
```
for layer in vgg.layers:
    layer.trainable = False
```

In [14]:
```
folders = glob('D:/DL/Covid-19/Covid-New/training_set/*')
x = Flatten()(vgg.output)
```

In [15]:
```
prediction = Dense(len(folders), activation='softmax')(x)
# create a model object
model = Model(inputs=vgg.input, outputs=prediction)
# view the structure of the model
model.summary()
```

Performance of the set model is evaluated using training and test accuracy and loss parameters.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
```

```
_________________________________________________________________
dense (Dense)                (None, 4)                 100356
=================================================================
Total params: 14,815,044
Trainable params: 100,356
Non-trainable params: 14,714,688
_________________________________________________
```

Adam optimizer is used to reach to the global minima while training out model. This optimizer will help us to toggle between local minima and global minima based on the settings. We have used 10 epochs with an iteration size of 600 per epoch.

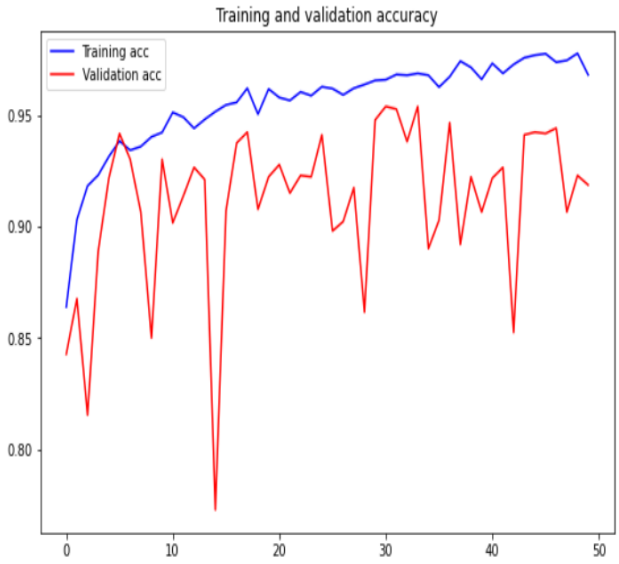
Total number of images in covid and normal folder in training data set is 6000 and number batches 10. So, the iteration for each epoch is 600.

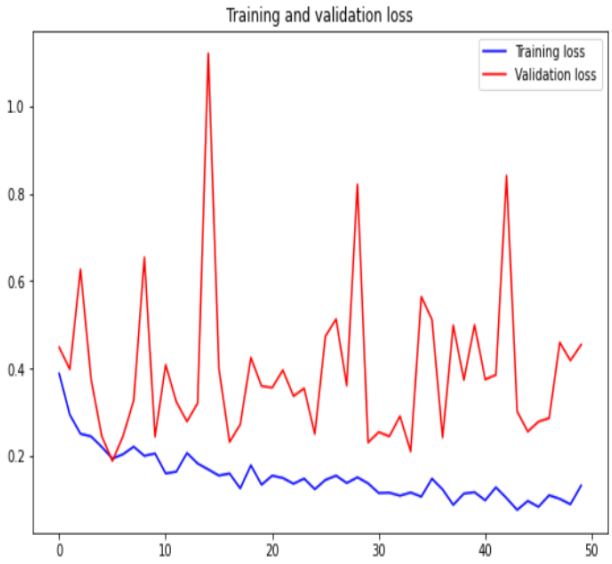
```
Epoch 40/50
600/600 [==============================] - 1493s 2s/step - loss: 0.1165 - accu
racy: 0.9660 - val_loss: 0.4994 - val_accuracy: 0.9064
Epoch 41/50
600/600 [==============================] - 1506s 3s/step - loss: 0.0978 - accu
racy: 0.9732 - val_loss: 0.3749 - val_accuracy: 0.9216
Epoch 42/50
600/600 [==============================] - 1491s 2s/step - loss: 0.1277 - accu
racy: 0.9687 - val_loss: 0.3850 - val_accuracy: 0.9265
Epoch 43/50
600/600 [==============================] - 1487s 2s/step - loss: 0.1030 - accu
racy: 0.9727 - val_loss: 0.8412 - val_accuracy: 0.8524
Epoch 44/50
600/600 [==============================] - 1512s 3s/step - loss: 0.0760 - accu
racy: 0.9757 - val_loss: 0.3012 - val_accuracy: 0.9411
Epoch 45/50
600/600 [==============================] - 1512s 3s/step - loss: 0.0965 - accu
racy: 0.9768 - val_loss: 0.2554 - val_accuracy: 0.9423
Epoch 46/50
600/600 [==============================] - 1500s 3s/step - loss: 0.0830 - accu
racy: 0.9775 - val_loss: 0.2781 - val_accuracy: 0.9417
Epoch 47/50
600/600 [==============================] - 1490s 2s/step - loss: 0.1096 - accu
racy: 0.9737 - val_loss: 0.2860 - val_accuracy: 0.9441
Epoch 48/50
600/600 [==============================] - 1491s 2s/step - loss: 0.1012 - accu
racy: 0.9745 - val_loss: 0.4593 - val_accuracy: 0.9064
```

8. Analysis and Preliminary Result

Model started has an incremental growth from 10th epoch till 47th epoch. From 40th epoch model has reached a state to yield 97% accuracy and from 47th epoch it is observed that there is loss in accuracy. Also validation accuracy is reached to 92% in 40th epoch.

```
In [21]: def plot_results(r):
             acc = r.history['accuracy']
             val_acc = r.history['val_accuracy']
             loss = r.history['loss']
             val_loss = r.history['val_loss']

             epoch = range(50)

             fig = plt.figure(figsize=(20,6))
             ax1 = fig.add_subplot(121)
             plt.plot(epoch, acc, 'b', label='Training acc')
             plt.plot(epoch, val_acc, 'r', label='Validation acc')
             ax1.set_title('Training and validation accuracy')
             ax1.legend()

             ax2 = fig.add_subplot(122)
             plt.plot(epoch, loss, 'b', label='Training loss')
             plt.plot(epoch, val_loss, 'r', label='Validation loss')
             ax2.set_title('Training and validation loss')
             ax2.legend()

             plt.show()

         plot_results(r)
```



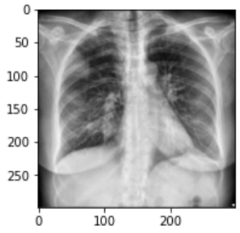


9. Test Results and Summary

Model was run with 30 samples to validate the prediction with out labels. Results are summarized in the following table.

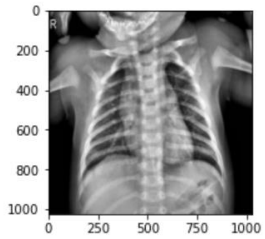
```
In [400]: img1=image.load_img('D:/DL/Covid-19/Prediction/X9.png',target_size=(224,224))

In [401]: x=image.img_to_array(img1)

In [402]: x=np.expand_dims(x, axis=0)
```

```
In [406]: img = cv2.imread('D:/DL/Covid-19/Prediction/X9.png')
          img = exposure.equalize_adapthist(img/np.max(img))
          plt.figure(figsize = (3,3))
          plt.imshow(img, 'gray')
          if result == 0:
              print("Person is Affected by Covid-19")
          else:
              print("Normal")

Person is Affected by Covid-19
```




```
In [422]: img=image.load_img('D:/DL/Covid-19/Prediction/N10.png',target_size=(224,224))

In [423]: x=image.img_to_array(img)

In [424]: x=np.expand_dims(x, axis=0)

In [425]: img_data=preprocess_input(x)

In [426]: classes=model.predict(img_data)

In [427]: result=int(classes[0][1])
          Y=result
          print(Y)

1
```

```
In [428]: img = cv2.imread('D:/DL/Covid-19/Prediction/N10.png')
          img = exposure.equalize_adapthist(img/np.max(img))
          plt.figure(figsize = (3,3))
          plt.imshow(img, 'gray')

          if Y==0:
              print("Person is Affected by Covid-19")
          else:
              print("Normal")

Normal
```



Prediction Image Number	Actual Value	Model Prediction	Status	Accuracy
X1	COVID-19	COVID-19	TRUE	100%
X2	COVID-19	COVID-19	TRUE	
X3	COVID-19	COVID-19	TRUE	
X4	COVID-19	COVID-19	TRUE	
X5	COVID-19	COVID-19	TRUE	
X6	COVID-19	COVID-19	TRUE	
X7	COVID-19	COVID-19	TRUE	
X8	COVID-19	COVID-19	TRUE	
X9	COVID-19	COVID-19	TRUE	
X10	COVID-19	COVID-19	TRUE	
N1	Normal	Normal	TRUE	100%
N2	Normal	Normal	TRUE	
N3	Normal	Normal	TRUE	
N4	Normal	Normal	TRUE	
N5	Normal	Normal	TRUE	
N6	Normal	Normal	TRUE	
N7	Normal	Normal	TRUE	
N8	Normal	Normal	TRUE	
N9	Normal	Normal	TRUE	
N10	Normal	Normal	TRUE	
P1	Viral Pneumonia	Viral Pneumonia	TRUE	70%
P2	Viral Pneumonia	Viral Pneumonia	TRUE	
P3	Viral Pneumonia	COVID-19	FALSE	
P4	Viral Pneumonia	Viral Pneumonia	TRUE	
P5	Viral Pneumonia	Viral Pneumonia	TRUE	
P6	Viral Pneumonia	Viral Pneumonia	TRUE	
P7	Viral Pneumonia	Viral Pneumonia	TRUE	
P8	Viral Pneumonia	Viral Pneumonia	TRUE	
P9	Viral Pneumonia	COVID-19	FALSE	
P10	Viral Pneumonia	COVID-19	FALSE	

10. Conclusion

1. This model is effectively detecting Covid and normal images with 100% accuracy.

2. While we introduce 3rd class Viral Pneumonia, Model has the capability to predict the Viral pneumonia with an accuracy of 70%, remaining 30% of times model is wrongly classifying viral pneumonia as Covid-19.