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

Task 1 [Total points: 30]

Setup

- This assignment involves the following packages: 'matplotlib', 'numpy', and 'sklearn'.
- If you are using conda, use the following commands to install the above packages:

```
conda install matplotlib
conda install numpy
conda install -c anaconda scikit-learn
```

- If you are using pip, use use the following commands to install the above packages:

```
pip install matplotlib
pip install numpy
pip install sklearn
```

Data

Please download the data using the following link: COVID-19.

- After downloading 'Covid_Data_GradientCrescent.zip', unzip the file and you should see the following data structure:

|--all
|-------train
|-------test
|--two
|-------train
|-------test

- Put the 'all' folder, the 'two' folder and this python notebook in the **same directory** so that the following code can correctly locate the data.

[20 points] Binary Classification: COVID-19 vs. Normal

```
import os

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

os.environ['OMP_NUM_THREADS'] = '1'
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
tf.__version__
```

```
'2.8.0'
```

Load Image Data

```
from google.colab import drive

drive.mount('/content/drive')

DATA_LIST = os.listdir('/content/drive/My Drive/CC/Covid_Data_GradientCrescent/'
DATASET_PATH  = '/content/drive/My Drive/CC/Covid_Data_GradientCrescent/two/trai
TEST_DIR =  '/content/drive/My Drive/CC/Covid_Data_GradientCrescent/two/test'

IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 10  # try reducing batch size or freeze more layers if your GPU
NUM_EPOCHS    = 40
LEARNING_RATE = 0.001 # start off with high rate first 0.001 and experiment with
print("done")
```

```
Mounted at /content/drive
done
```

Generate Training and Validation Batches

```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_
                                   featurewise_std_normalization = True,width_sh
                                   height_shift_range=0.2,shear_range=0.25,zoom_
                                   zca_whitening = True,channel_shift_range = 20
                                   horizontal_flip = True,vertical_flip = True,
                                   validation_split = 0.2,fill_mode='constant')

train_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE
                                                  shuffle=True,batch_size=BATCH_
                                                  subset = "training",seed=42,
                                                  class_mode="binary")

valid_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE
                                                  shuffle=True,batch_size=BATCH_
                                                  subset = "validation",seed=42,
                                                  class_mode="binary")
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data
  warnings.warn('This ImageDataGenerator specifies '
Found 104 images belonging to 2 classes.
Found 26 images belonging to 2 classes.
```

[10 points] Build Model

Hint: Starting from a pre-trained model typically helps performance on a new task, e.g. starting with weights obtained by training on ImageNet.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras import regularizers
#from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import numpy as np
import argparse
```

```
model = tf.keras.models.Sequential()
model.add (VGG16 (weights= 'imagenet', include_top=False, input_shape =(224,224,
model.add (Flatten())
model.add (BatchNormalization())
model.add(Dense(units=256,activation="relu"))
model.add(Dropout(0.25))
model.add(Dense(units=1,activation="sigmoid"))
model.layers[0].trainable = False
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688  

 flatten (Flatten)           (None, 25088)             0         

 batch_normalization (BatchN  (None, 25088)            100352    
 ormalization)                                                   

 dense (Dense)               (None, 256)               6422784   

 dropout (Dropout)           (None, 256)               0         

 dense_1 (Dense)             (None, 1)                 257       

=================================================================
Total params: 21,238,081
Trainable params: 6,473,217
Non-trainable params: 14,764,864
_________________________________________________________________
```

[5 points] Train Model

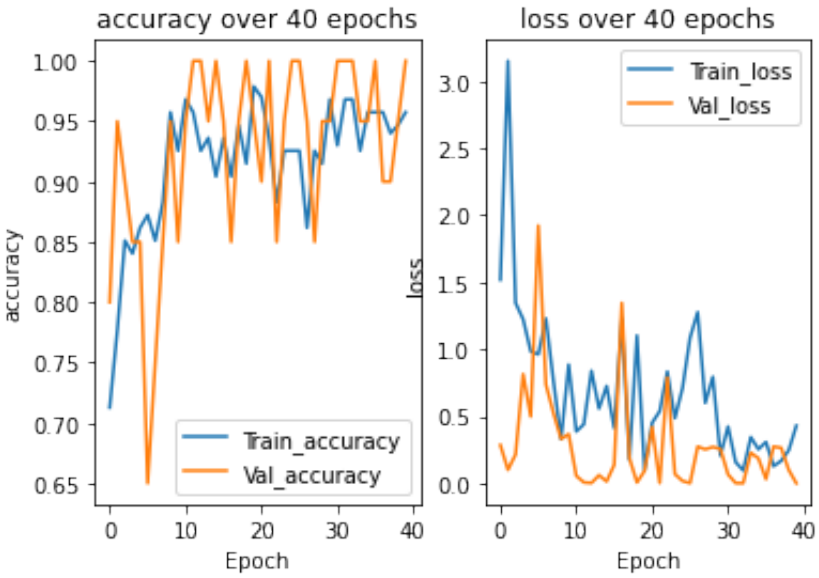
```
#FIT MODEL
print(len(train_batches))
print(len(valid_batches))

STEP_SIZE_TRAIN=train_batches.n//train_batches.batch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']
history = model.fit(x=train_batches,epochs=NUM_EPOCHS,steps_per_epoch=STEP_SIZE_
```

```
      validation_steps=STEP_SIZE_VALID)
Epoch 11/40
10/10 [==============================] - 50s 5s/step - loss: 0.3871 - accur
Epoch 12/40
10/10 [==============================] - 50s 5s/step - loss: 0.4407 - accur
Epoch 13/40
10/10 [==============================] - 50s 5s/step - loss: 0.8392 - accur
Epoch 14/40
10/10 [==============================] - 51s 5s/step - loss: 0.5575 - accur
Epoch 15/40
10/10 [==============================] - 51s 5s/step - loss: 0.7256 - accur
Epoch 16/40
10/10 [==============================] - 50s 5s/step - loss: 0.4122 - accur
Epoch 17/40
10/10 [==============================] - 51s 5s/step - loss: 1.2241 - accur
Epoch 18/40
10/10 [==============================] - 51s 5s/step - loss: 0.1749 - accur
Epoch 19/40
10/10 [==============================] - 51s 5s/step - loss: 1.1028 - accur
Epoch 20/40
10/10 [==============================] - 50s 5s/step - loss: 0.1011 - accur
Epoch 21/40
10/10 [==============================] - 53s 5s/step - loss: 0.4513 - accur
Epoch 22/40
10/10 [==============================] - 50s 5s/step - loss: 0.5381 - accur
Epoch 23/40
10/10 [==============================] - 50s 5s/step - loss: 0.8335 - accur
Epoch 24/40
10/10 [==============================] - 50s 5s/step - loss: 0.4860 - accur
Epoch 25/40
10/10 [==============================] - 50s 5s/step - loss: 0.7099 - accur
Epoch 26/40
10/10 [==============================] - 50s 5s/step - loss: 1.0873 - accur
Epoch 27/40
10/10 [==============================] - 51s 5s/step - loss: 1.2768 - accur
Epoch 28/40
10/10 [==============================] - 51s 5s/step - loss: 0.5998 - accur
Epoch 29/40
10/10 [==============================] - 52s 5s/step - loss: 0.7942 - accur
Epoch 30/40
10/10 [==============================] - 51s 5s/step - loss: 0.2068 - accur
Epoch 31/40
10/10 [==============================] - 54s 5s/step - loss: 0.4232 - accur
Epoch 32/40
10/10 [==============================] - 51s 5s/step - loss: 0.1580 - accur
Epoch 33/40
10/10 [==============================] - 51s 5s/step - loss: 0.0960 - accur
Epoch 34/40
10/10 [==============================] - 50s 5s/step - loss: 0.3441 - accur
Epoch 35/40
10/10 [==============================] - 51s 5s/step - loss: 0.2539 - accur
Epoch 36/40
10/10 [==============================] - 50s 5s/step - loss: 0.3076 - accur
Epoch 37/40
```

```
10/10 [==============================] - 50s 5s/step - loss: 0.1277 - accur
Epoch 38/40
10/10 [==============================] - 53s 5s/step - loss: 0.1674 - accur
Epoch 39/40
10/10 [==============================] - 50s 5s/step - loss: 0.2462 - accur
Epoch 40/40
10/10 [==============================] - 50s 5s/step - loss: 0.4294 - accur
```

[5 points] Plot Accuracy and Loss During Training

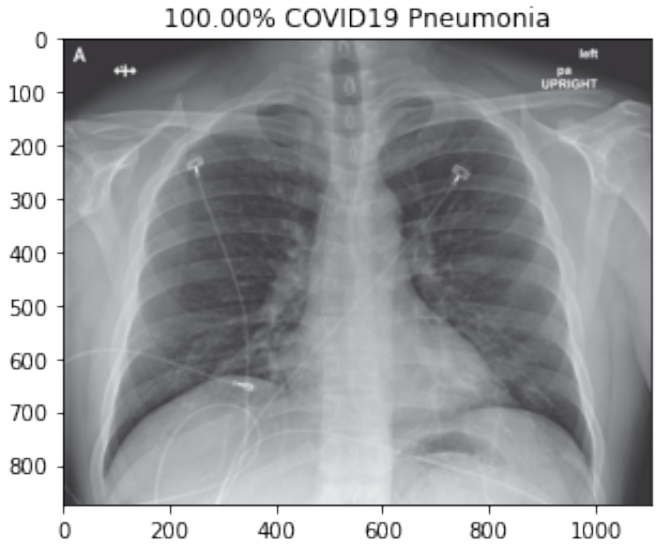
```
import matplotlib.pyplot as plt

fig, (ax) = plt.subplots(1, 2)

c=0

for i in ['accuracy', 'loss']:
  ax[c].plot(history.history[i], label='Train_'+i)
  ax[c].plot(history.history['val_'+i], label='Val_'+i)
  ax[c].set_xlabel('Epoch')
  ax[c].set_ylabel(i)
  if i=='accuracy':
    ax[c].legend(loc='lower right')
  else:
    ax[c].legend(loc='upper right')

  ax[c].set_title(str(i)+' over '+str(NUM_EPOCHS)+' epochs')
  c+=1

plt.show()
```



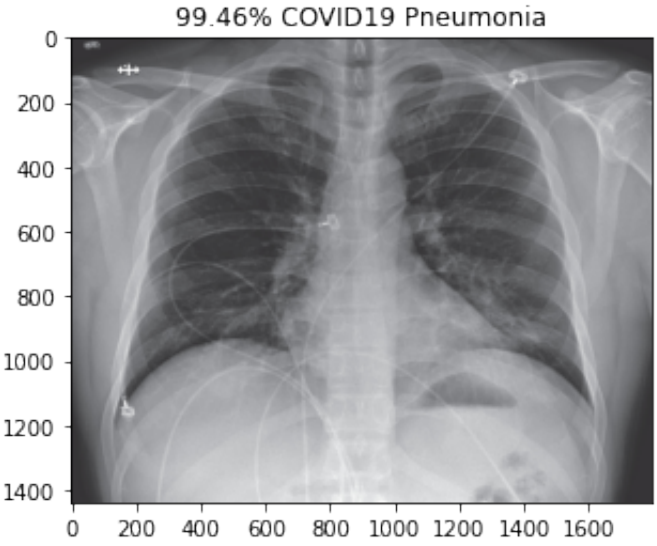
Plot Test Results

```
import matplotlib.image as mpimg

test_datagen = ImageDataGenerator(rescale=1. / 255)
eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZ
                                                  batch_size=1,shuffle=True,seed
eval_generator.reset()
pred = model.predict_generator(eval_generator,18,verbose=1)
for index, probability in enumerate(pred):
    image_path = TEST_DIR + "/" +eval_generator.filenames[index]
    image = mpimg.imread(image_path)
    if image.ndim < 3:
        image = np.reshape(image,(image.shape[0],image.shape[1],1))
        image = np.concatenate([image, image, image], 2)
#         print(image.shape)

    pixels = np.array(image)
    plt.imshow(pixels)

    print(eval_generator.filenames[index])
    if probability > 0.5:
        plt.title("%.2f" % (probability[0]*100) + "% Normal")
    else:
        plt.title("%.2f" % ((1-probability[0])*100) + "% COVID19 Pneumonia")
    plt.show()
```

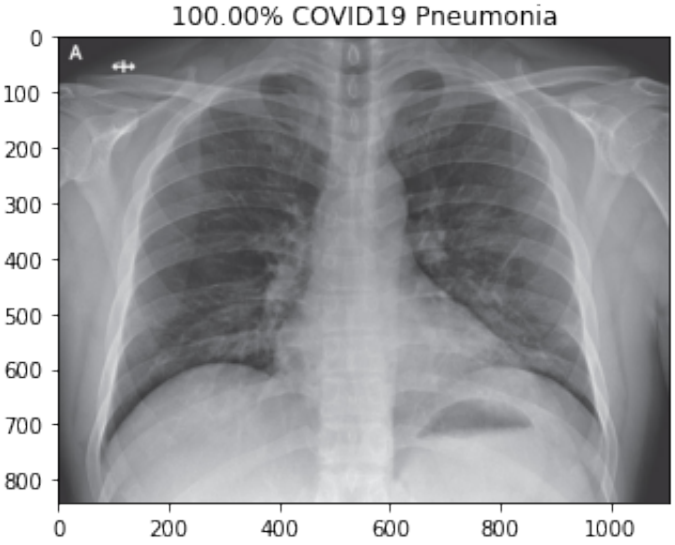
```
Found 18 images belonging to 2 classes.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: UserWarning
  import sys
18/18 [==============================] - 9s 495ms/step
covid/nejmoa2001191_f3-PA.jpeg
```



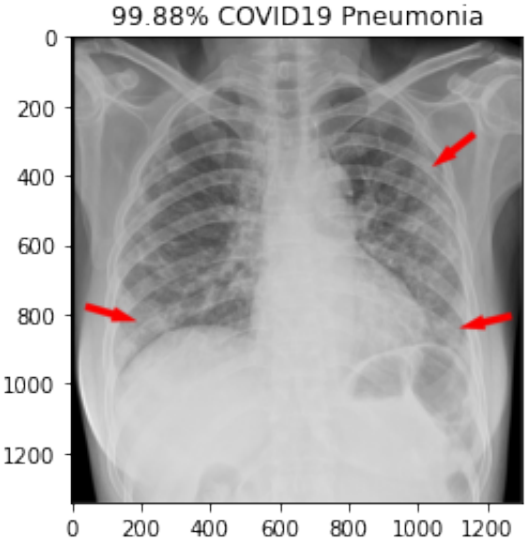

```
covid/nejmoa2001191_f4.jpeg
```



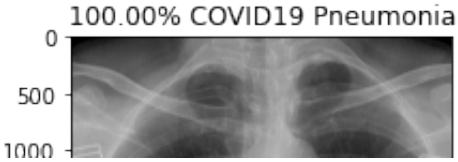

```
covid/nejmoa2001191_f5-PA.jpeg
```

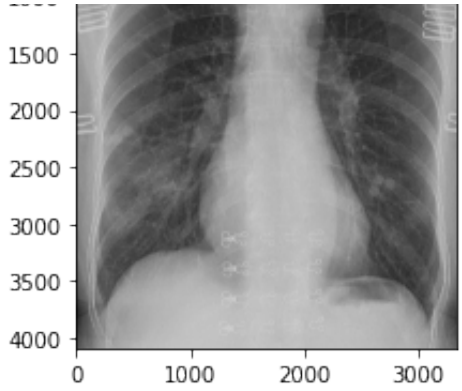



```
covid/radiol.2020200490.fig3.jpeg
```

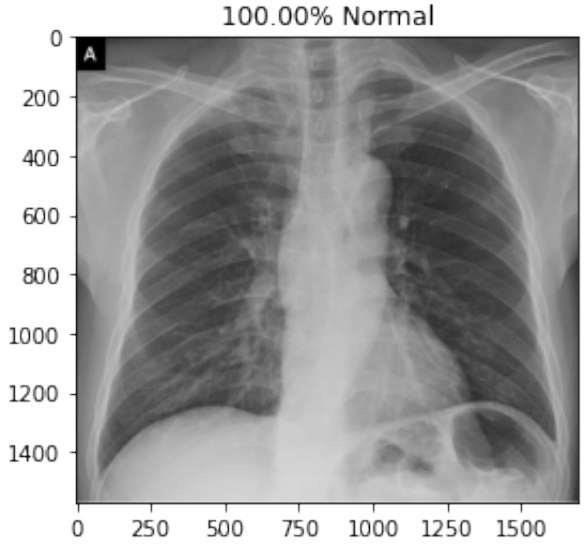



```
covid/ryct.2020200028.fig1a.jpeg
```

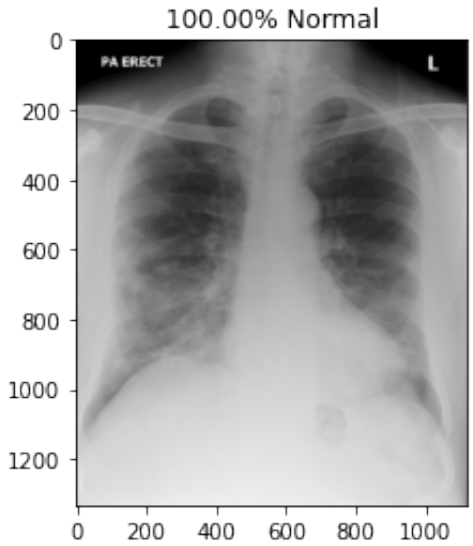




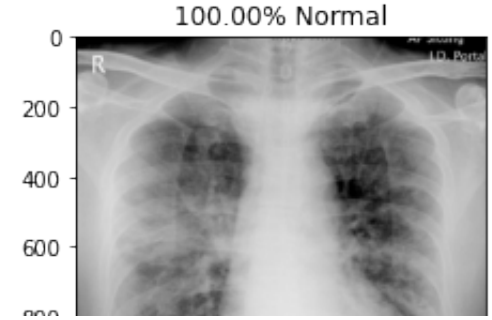
covid/ryct.2020200034.fig2.jpeg

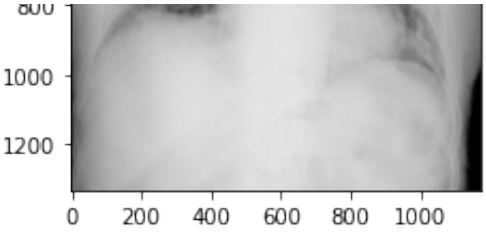


covid/ryct.2020200034.fig5-day0.jpeg

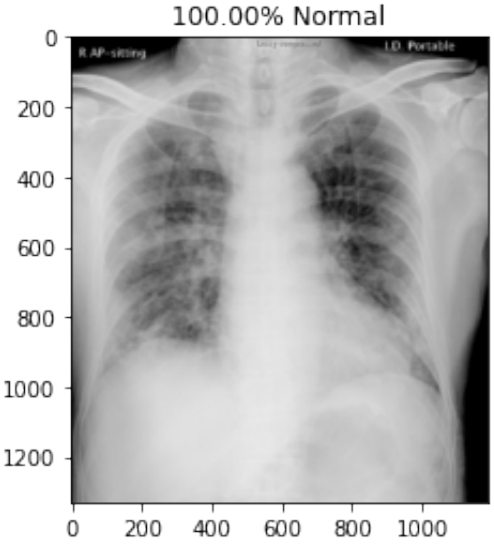


covid/ryct.2020200034.fig5-day4.jpeg

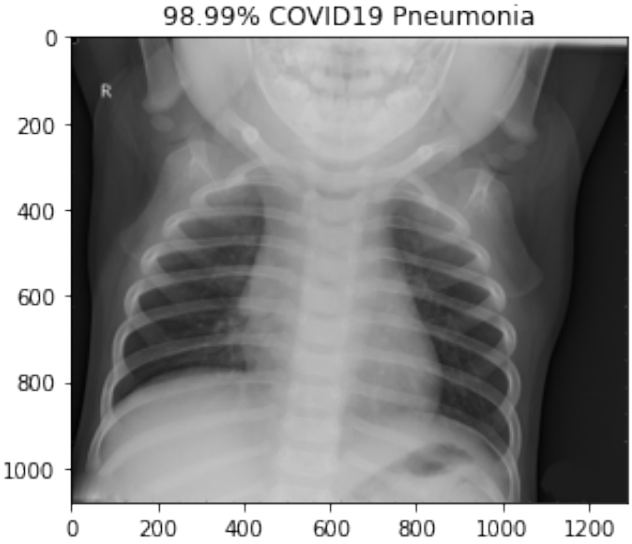




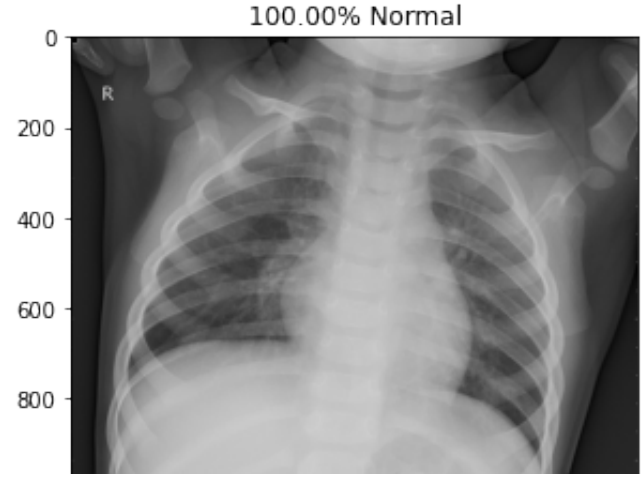
covid/ryct.2020200034.fig5-day7.jpeg

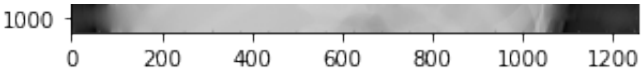


normal/NORMAL2-IM-1385-0001.jpeg

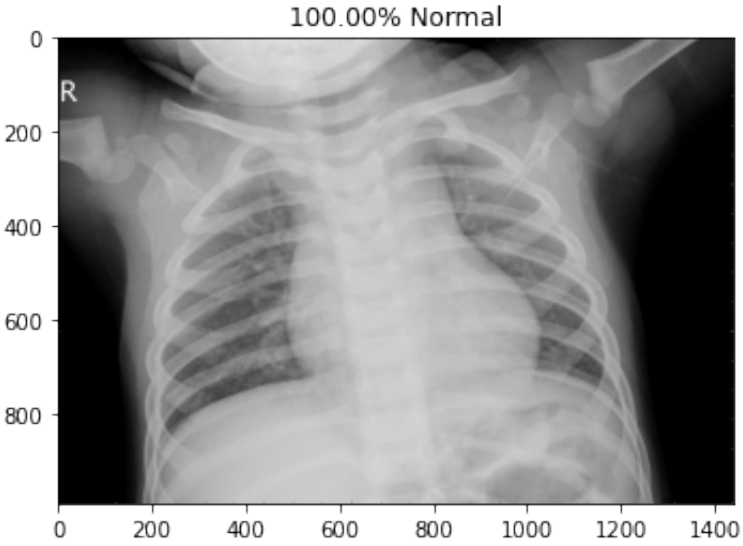


normal/NORMAL2-IM-1396-0001.jpeg

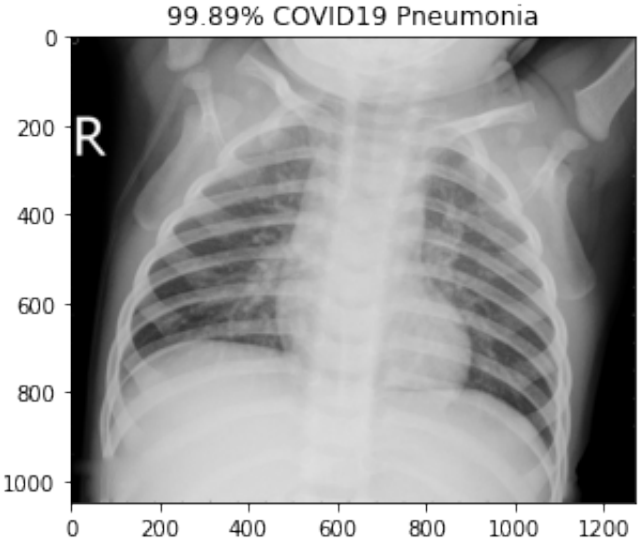




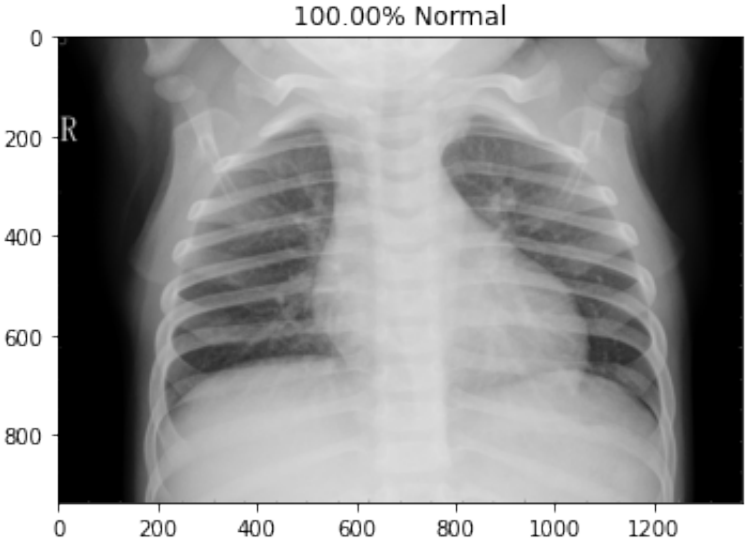
normal/NORMAL2-IM-1400-0001.jpeg



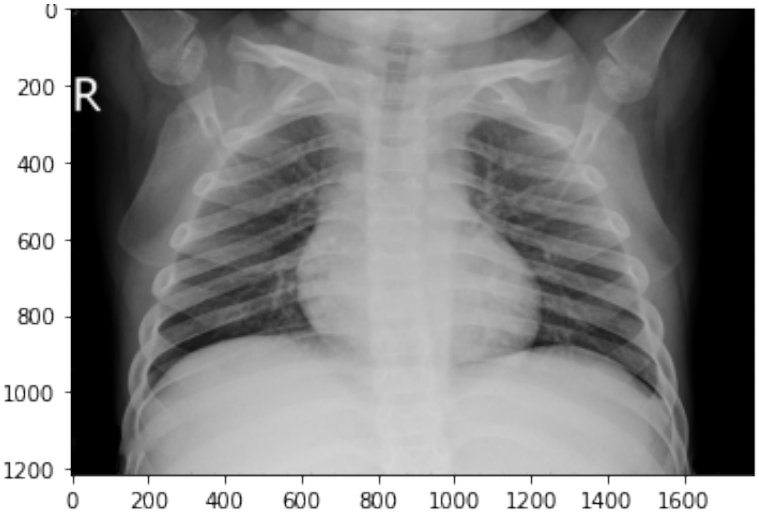
normal/NORMAL2-IM-1401-0001.jpeg



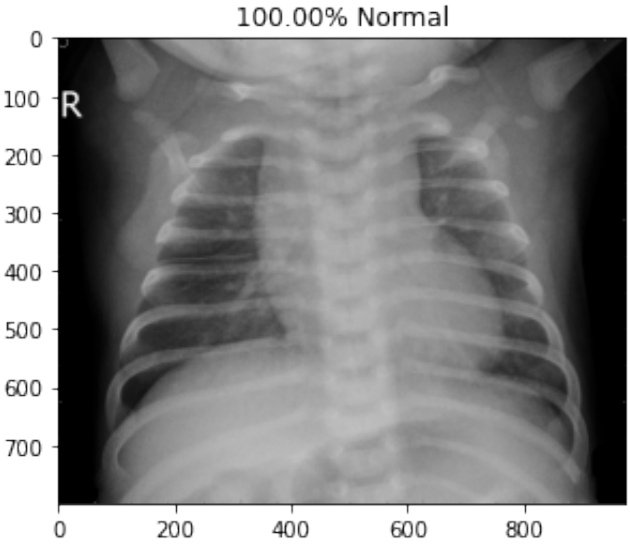
normal/NORMAL2-IM-1406-0001.jpeg



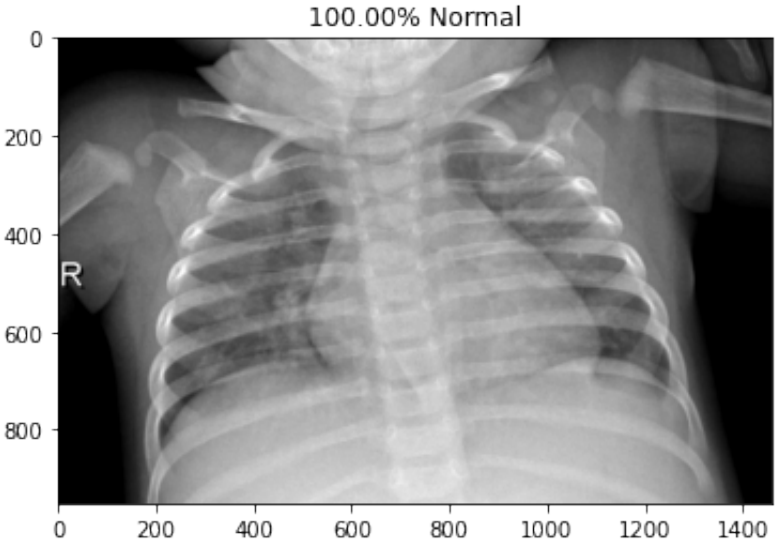
normal/NORMAL2-IM-1412-0001.jpeg



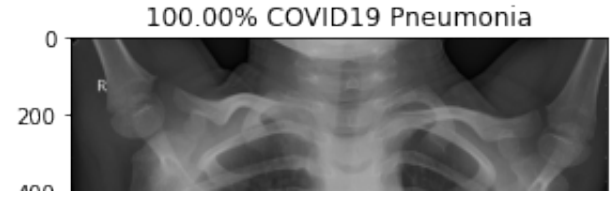
normal/NORMAL2-IM-1419-0001.jpeg

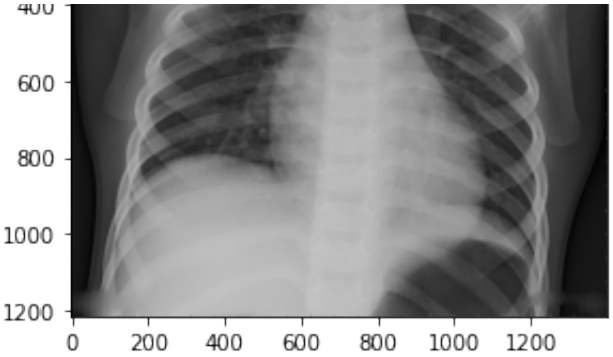


normal/NORMAL2-IM-1422-0001.jpeg



normal/NORMAL2-IM-1423-0001.jpeg





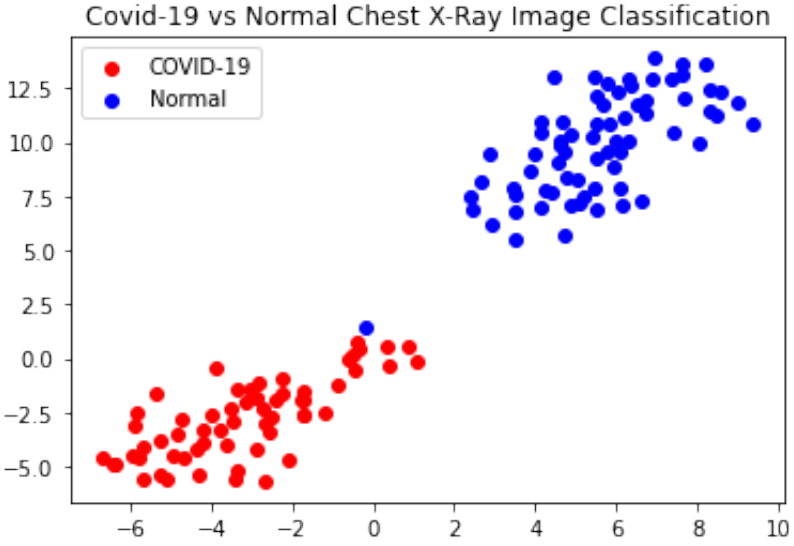
[10 points] TSNE Plot

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely used technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. After training is complete, extract features from a specific deep layer of your choice, use t-SNE to reduce the dimensionality of your extracted features to 2 dimensions and plot the resulting 2D features.

```
from sklearn.manifold import TSNE
from tensorflow.keras import models

intermediate_layer_model = models.Model(inputs=model.input, outputs=model.get_l
tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size
                                                       batch_size=1,shuffle=False,se

activations = intermediate_layer_model.predict(tsne_data_generator, verbose = 1
tsne = TSNE(n_components=2)
tsne_obj = tsne.fit_transform(activations)

x1 = []
x2 = []
y1 = []
y2 = []



for i in range(tsne_obj.shape[0]):
    if(tsne_data_generator.labels[i]==0):
        x1.append(tsne_obj[i,0])
        y1.append(tsne_obj[i,1])
    else:

        x2.append(tsne_obj[i,1])
        y2.append(tsne_obj[i,0])
```

```
plt.scatter(x1,y1, c="red", label="COVID-19")
plt.scatter(x2,y2, c="blue", label="Normal")
plt.title('Covid-19 vs Normal Chest X-Ray Image Classification')
plt.legend()
plt.show()
```

```
Found 130 images belonging to 2 classes.
130/130 [==============================] - 63s 477ms/step
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:783: Futu
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: Futu
  FutureWarning,
```



✓ 1m 23s completed at 10:16 PM