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 (https://drive.google.com/file/d/1Y88tgqpQ1Pjko_7rntcPowOJs_QNOrJ-/view).

- 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

In [1]:

```
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_DEVICE_ORDER"]="PCI_BUS_ID"
# os.environ['CUDA_VISIBLE_DEVICES'] = '2'
tf.__version__
```

Out[1]:

```
'2.6.0'
```

In [2]:

```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 7688805591285904748
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 6252920832
locality {
  bus_id: 1
  links {
  }
}
incarnation: 1699347087621046735
physical_device_desc: "device: 0, name: NVIDIA GeForce RTX 2080 SUPE
R, pci bus id: 0000:01:00.0, compute capability: 7.5"
]
```

Load Image Data

In [3]:

```
DATA_LIST = os.listdir('two/train')
DATASET_PATH  = 'two/train'
TEST_DIR =  '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 runs out of memory
NUM_EPOCHS    = 40
LEARNING_RATE = 0.0005 # start off with high rate first 0.001 and experiment with reducing it gradually
```

Generate Training and Validation Batches

In [4]:

```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center = True,
                                   featurewise_std_normalization = True,width_shift_range=0.2,
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0.1,
                                   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_SIZE,
                                                  shuffle=True,batch_size=BATCH_SIZE,
                                                  subset = "training",seed=42,
                                                  class_mode="binary")

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

```
Found 104 images belonging to 2 classes.
Found 26 images belonging to 2 classes.

C:\Users\Li\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.py:342: UserWarning: This ImageDataGenerator specifies `zca_whitening` which overrides setting of`featurewise_std_normalization`.
  warnings.warn('This ImageDataGenerator specifies '
```

[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.

In [5]:

```
from tensorflow.keras.applications import VGG16
from keras.layers.core import Flatten, Dense, Dropout, Lambda


vgg16 = VGG16(weights='imagenet',include_top=False,pooling = "None", classes = 2
, input_shape=(224, 224, 3))
vgg16.trainable = False

model = tf.keras.models.Sequential()
model.add(vgg16)
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.summary()

model.compile(loss='binary_crossentropy', optimizer=tf.keras.optimizers.Adam(lea
rning_rate=LEARNING_RATE), metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 256)               6422784
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,137,729
Trainable params: 6,423,041
Non-trainable params: 14,714,688
_________________________________________________________________
```

[5 points] Train Model

In [6]:

```
#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

import time
start = time.time()

history=model.fit_generator(train_batches, steps_per_epoch =STEP_SIZE_TRAIN, val
idation_data = valid_batches, validation_steps = STEP_SIZE_VALID, epochs= NUM_EP
OCHS)

end = time.time()
total_time = end - start
```

```
11
3

C:\Users\Li\anaconda3\lib\site-packages\keras\engine\training.py:197
2: UserWarning: `Model.fit_generator` is deprecated and will be remo
ved in a future version. Please use `Model.fit`, which supports gene
rators.
  warnings.warn('`Model.fit_generator` is deprecated and '
C:\Users\Li\anaconda3\lib\site-packages\keras_preprocessing\image\im
age_data_generator.py:720: UserWarning: This ImageDataGenerator spec
ifies `featurewise_center`, but it hasn't been fit on any training d
ata. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
C:\Users\Li\anaconda3\lib\site-packages\keras_preprocessing\image\im
age_data_generator.py:739: UserWarning: This ImageDataGenerator spec
ifies `zca_whitening`, but it hasn't been fit on any training data.
Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '

Epoch 1/40
10/10 [==============================] - 6s 340ms/step - loss: 2.369
0 - accuracy: 0.4149 - val_loss: 0.7184 - val_accuracy: 0.5500
Epoch 2/40
10/10 [==============================] - 3s 299ms/step - loss: 0.652
5 - accuracy: 0.5638 - val_loss: 0.5157 - val_accuracy: 0.9500
Epoch 3/40
10/10 [==============================] - 3s 306ms/step - loss: 0.592
8 - accuracy: 0.6383 - val_loss: 0.3904 - val_accuracy: 0.9500
Epoch 4/40
10/10 [==============================] - 3s 307ms/step - loss: 0.462
8 - accuracy: 0.8298 - val_loss: 0.4797 - val_accuracy: 0.7500
Epoch 5/40
10/10 [==============================] - 3s 288ms/step - loss: 0.398
3 - accuracy: 0.8191 - val_loss: 0.2820 - val_accuracy: 0.9500
Epoch 6/40
```

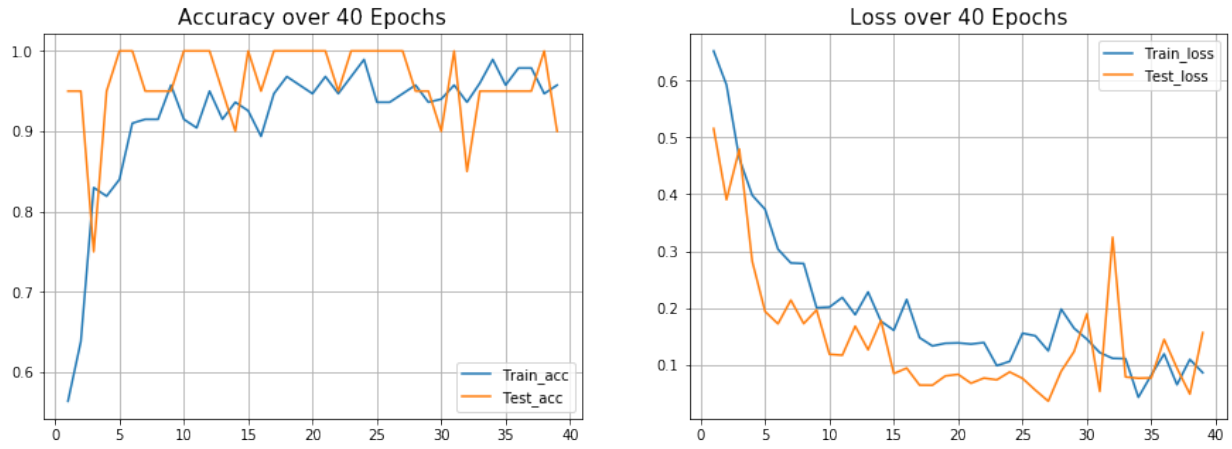
```
10/10 [==============================] - 3s 325ms/step - loss: 0.373
3 - accuracy: 0.8400 - val_loss: 0.1934 - val_accuracy: 1.0000
Epoch 7/40
10/10 [==============================] - 3s 326ms/step - loss: 0.303
2 - accuracy: 0.9100 - val_loss: 0.1718 - val_accuracy: 1.0000
Epoch 8/40
10/10 [==============================] - 3s 312ms/step - loss: 0.279
0 - accuracy: 0.9149 - val_loss: 0.2136 - val_accuracy: 0.9500
Epoch 9/40
10/10 [==============================] - 3s 316ms/step - loss: 0.277
9 - accuracy: 0.9149 - val_loss: 0.1722 - val_accuracy: 0.9500
Epoch 10/40
10/10 [==============================] - 3s 309ms/step - loss: 0.200
1 - accuracy: 0.9574 - val_loss: 0.1962 - val_accuracy: 0.9500
Epoch 11/40
10/10 [==============================] - 3s 299ms/step - loss: 0.201
2 - accuracy: 0.9149 - val_loss: 0.1182 - val_accuracy: 1.0000
Epoch 12/40
10/10 [==============================] - 3s 310ms/step - loss: 0.218
2 - accuracy: 0.9043 - val_loss: 0.1165 - val_accuracy: 1.0000
Epoch 13/40
10/10 [==============================] - 3s 317ms/step - loss: 0.187
9 - accuracy: 0.9500 - val_loss: 0.1677 - val_accuracy: 1.0000
Epoch 14/40
10/10 [==============================] - 3s 330ms/step - loss: 0.227
8 - accuracy: 0.9149 - val_loss: 0.1260 - val_accuracy: 0.9500
Epoch 15/40
10/10 [==============================] - 3s 310ms/step - loss: 0.176
4 - accuracy: 0.9362 - val_loss: 0.1776 - val_accuracy: 0.9000
Epoch 16/40
10/10 [==============================] - 3s 312ms/step - loss: 0.160
4 - accuracy: 0.9255 - val_loss: 0.0842 - val_accuracy: 1.0000
Epoch 17/40
10/10 [==============================] - 3s 323ms/step - loss: 0.214
7 - accuracy: 0.8936 - val_loss: 0.0938 - val_accuracy: 0.9500
Epoch 18/40
10/10 [==============================] - 3s 281ms/step - loss: 0.147
4 - accuracy: 0.9468 - val_loss: 0.0638 - val_accuracy: 1.0000
Epoch 19/40
10/10 [==============================] - 3s 289ms/step - loss: 0.132
8 - accuracy: 0.9681 - val_loss: 0.0636 - val_accuracy: 1.0000
Epoch 20/40
10/10 [==============================] - 3s 299ms/step - loss: 0.137
5 - accuracy: 0.9574 - val_loss: 0.0799 - val_accuracy: 1.0000
Epoch 21/40
10/10 [==============================] - 3s 310ms/step - loss: 0.138
1 - accuracy: 0.9468 - val_loss: 0.0828 - val_accuracy: 1.0000
Epoch 22/40
10/10 [==============================] - 3s 276ms/step - loss: 0.136
1 - accuracy: 0.9681 - val_loss: 0.0671 - val_accuracy: 1.0000
Epoch 23/40
10/10 [==============================] - 3s 300ms/step - loss: 0.138
8 - accuracy: 0.9468 - val_loss: 0.0762 - val_accuracy: 0.9500
```

```
Epoch 24/40
10/10 [==============================] - 3s 300ms/step - loss: 0.098
1 - accuracy: 0.9681 - val_loss: 0.0731 - val_accuracy: 1.0000
Epoch 25/40
10/10 [==============================] - 3s 292ms/step - loss: 0.105
6 - accuracy: 0.9894 - val_loss: 0.0870 - val_accuracy: 1.0000
Epoch 26/40
10/10 [==============================] - 3s 297ms/step - loss: 0.155
2 - accuracy: 0.9362 - val_loss: 0.0756 - val_accuracy: 1.0000
Epoch 27/40
10/10 [==============================] - 3s 297ms/step - loss: 0.150
6 - accuracy: 0.9362 - val_loss: 0.0549 - val_accuracy: 1.0000
Epoch 28/40
10/10 [==============================] - 3s 292ms/step - loss: 0.124
1 - accuracy: 0.9468 - val_loss: 0.0355 - val_accuracy: 1.0000
Epoch 29/40
10/10 [==============================] - 3s 308ms/step - loss: 0.197
7 - accuracy: 0.9574 - val_loss: 0.0881 - val_accuracy: 0.9500
Epoch 30/40
10/10 [==============================] - 3s 311ms/step - loss: 0.164
0 - accuracy: 0.9362 - val_loss: 0.1227 - val_accuracy: 0.9500
Epoch 31/40
10/10 [==============================] - 3s 321ms/step - loss: 0.144
7 - accuracy: 0.9400 - val_loss: 0.1893 - val_accuracy: 0.9000
Epoch 32/40
10/10 [==============================] - 3s 301ms/step - loss: 0.121
1 - accuracy: 0.9574 - val_loss: 0.0527 - val_accuracy: 1.0000
Epoch 33/40
10/10 [==============================] - 3s 302ms/step - loss: 0.111
0 - accuracy: 0.9362 - val_loss: 0.3244 - val_accuracy: 0.8500
Epoch 34/40
10/10 [==============================] - 3s 323ms/step - loss: 0.110
4 - accuracy: 0.9600 - val_loss: 0.0782 - val_accuracy: 0.9500
Epoch 35/40
10/10 [==============================] - 3s 303ms/step - loss: 0.042
3 - accuracy: 0.9894 - val_loss: 0.0759 - val_accuracy: 0.9500
Epoch 36/40
10/10 [==============================] - 3s 311ms/step - loss: 0.081
5 - accuracy: 0.9574 - val_loss: 0.0766 - val_accuracy: 0.9500
Epoch 37/40
10/10 [==============================] - 3s 324ms/step - loss: 0.118
8 - accuracy: 0.9787 - val_loss: 0.1443 - val_accuracy: 0.9500
Epoch 38/40
10/10 [==============================] - 3s 304ms/step - loss: 0.064
6 - accuracy: 0.9787 - val_loss: 0.0936 - val_accuracy: 0.9500
Epoch 39/40
10/10 [==============================] - 3s 291ms/step - loss: 0.109
1 - accuracy: 0.9468 - val_loss: 0.0480 - val_accuracy: 1.0000
Epoch 40/40
10/10 [==============================] - 3s 314ms/step - loss: 0.085
7 - accuracy: 0.9574 - val_loss: 0.1562 - val_accuracy: 0.9000
```

[5 points] Plot Accuracy and Loss During Training

In [7]:

```
import matplotlib.pyplot as plt

def plot_acc_loss(history, epochs):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    plt.figure(figsize=(15, 5))

    plt.subplot(121)
    plt.plot(range(1,epochs), acc[1:], label='Train_acc')
    plt.plot(range(1,epochs), val_acc[1:], label='Test_acc')
    plt.title('Accuracy over ' + str(epochs) + ' Epochs', size=15)
    plt.grid(True)
    plt.legend()

    plt.subplot(122)
    plt.plot(range(1,epochs), loss[1:], label='Train_loss')
    plt.plot(range(1,epochs), val_loss[1:], label='Test_loss')
    plt.title('Loss over ' + str(epochs) + ' Epochs', size=15)
    plt.grid(True)
    plt.legend()

    plt.show()


plot_acc_loss(history, 40)
```

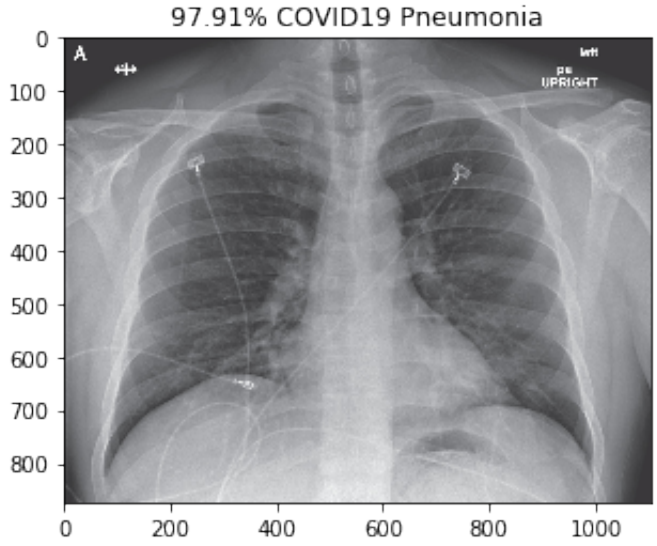


Plot Test Results

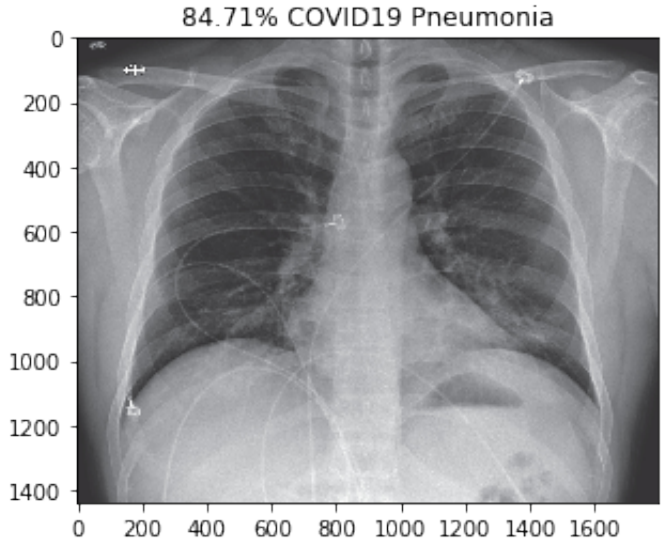
In [8]:

```
import matplotlib.image as mpimg

test_datagen = ImageDataGenerator(rescale=1. / 255)
eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZ
E,
                                                  batch_size=1,shuffle=True,seed
=42,class_mode="binary")
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()
```

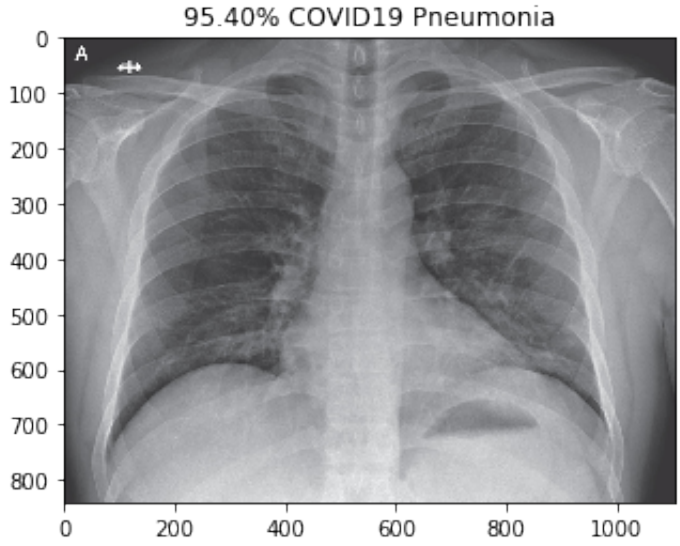
```
Found 18 images belonging to 2 classes.

C:\Users\Li\anaconda3\lib\site-packages\keras\engine\training.py:203
5: UserWarning: `Model.predict_generator` is deprecated and will be
removed in a future version. Please use `Model.predict`, which suppo
rts generators.
  warnings.warn('`Model.predict_generator` is deprecated and '

18/18 [==============================] - 1s 20ms/step
covid\nejmoa2001191_f3-PA.jpeg
```



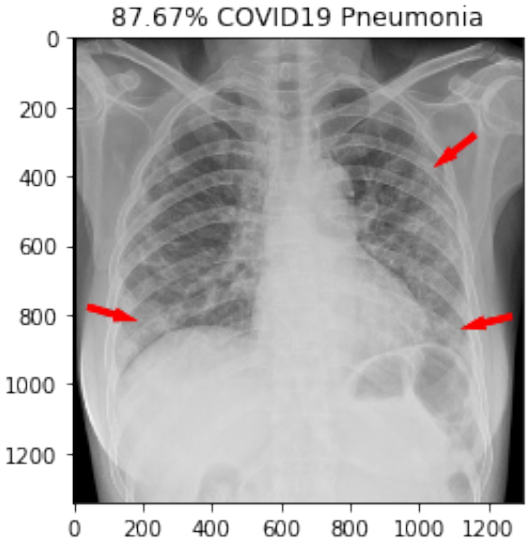
covid\nejmoa2001191_f4.jpeg



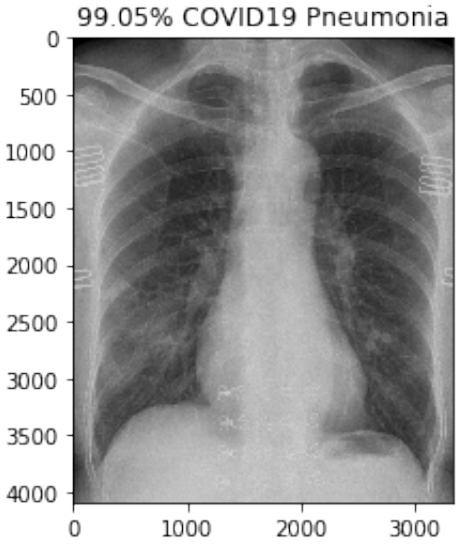
covid\nejmoa2001191_f5-PA.jpeg



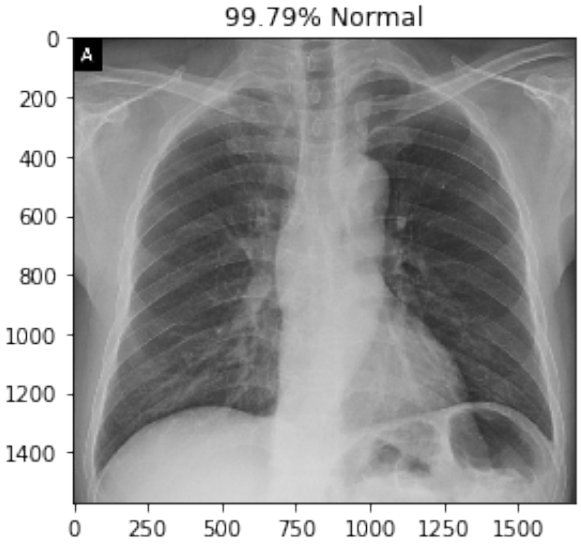
covid\radiol.2020200490.fig3.jpeg



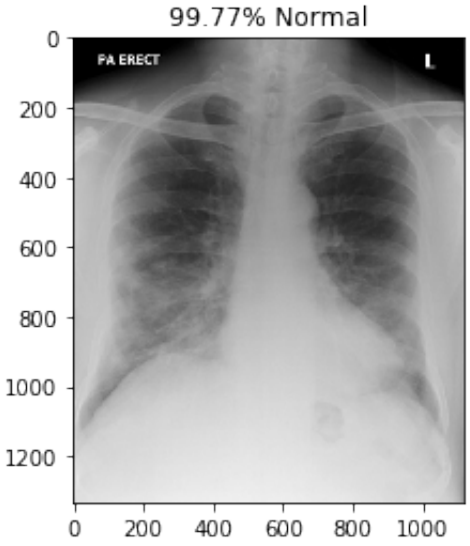
covid\ryct.2020200028.fig1a.jpeg



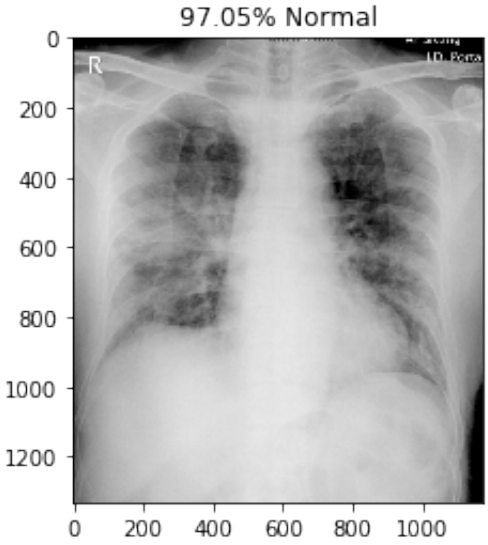
covid\ryct.2020200034.fig2.jpeg



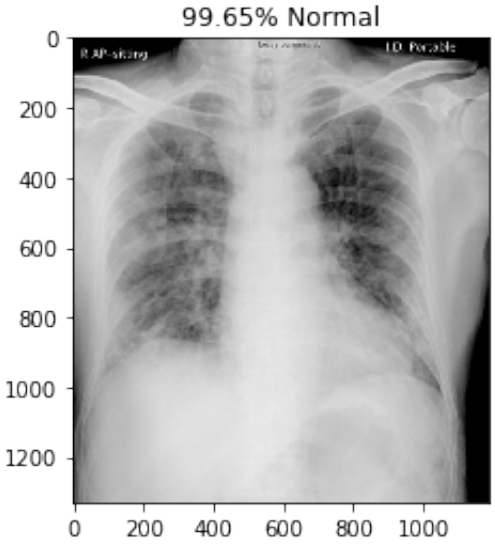
covid\ryct.2020200034.fig5-day0.jpeg



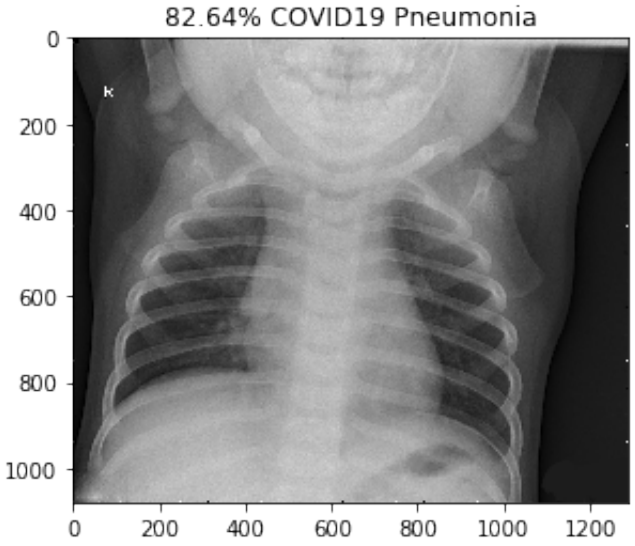
covid\ryct.2020200034.fig5-day4.jpeg



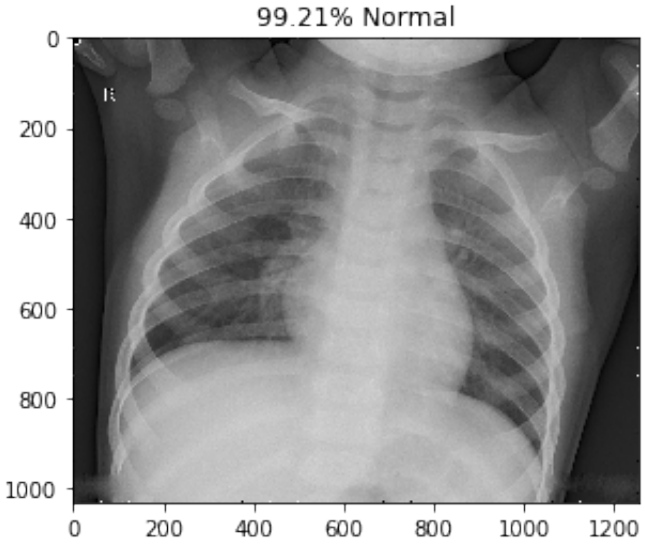
covid\ryct.2020200034.fig5-day7.jpeg



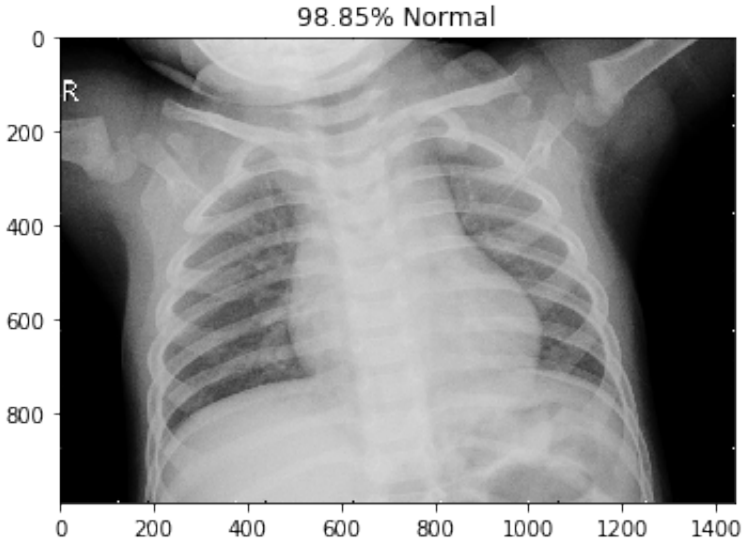
normal\NORMAL2-IM-1385-0001.jpeg



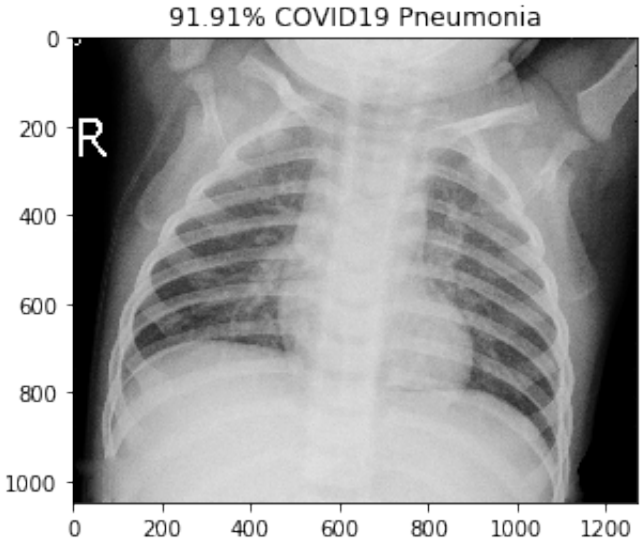
normal\NORMAL2-IM-1396-0001.jpeg



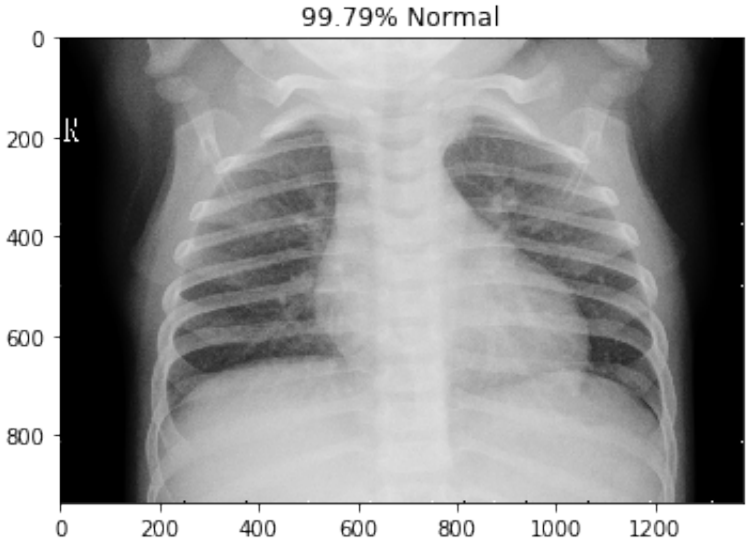
normal\NORMAL2-IM-1400-0001.jpeg



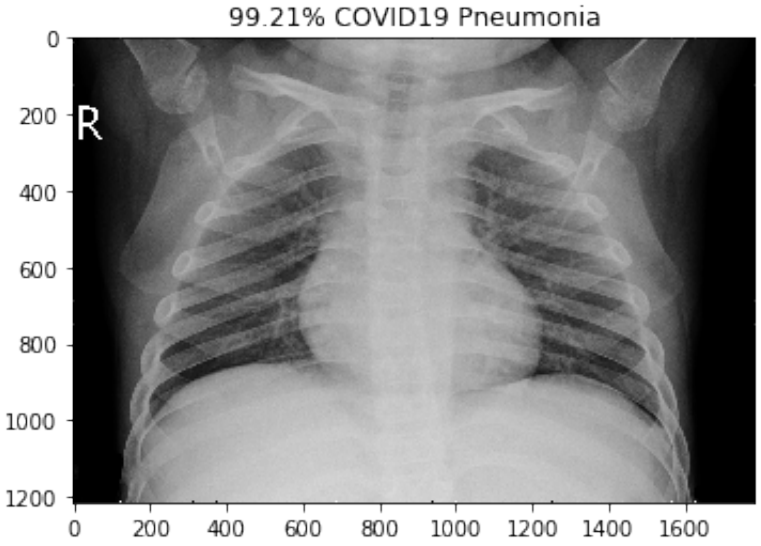
normal\NORMAL2-IM-1401-0001.jpeg



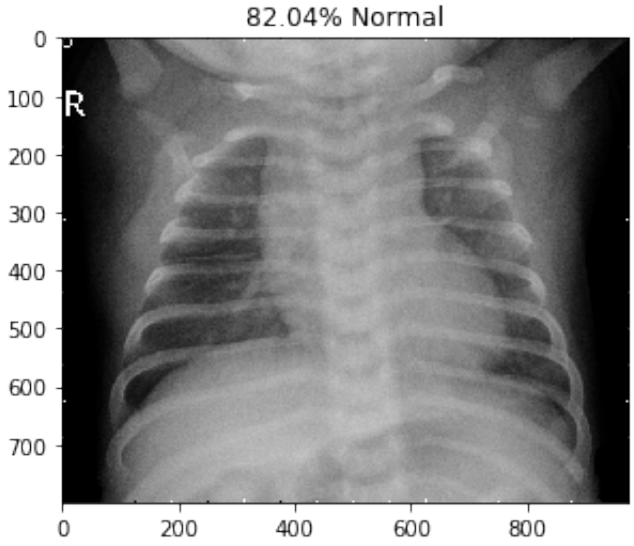
normal\NORMAL2-IM-1406-0001.jpeg



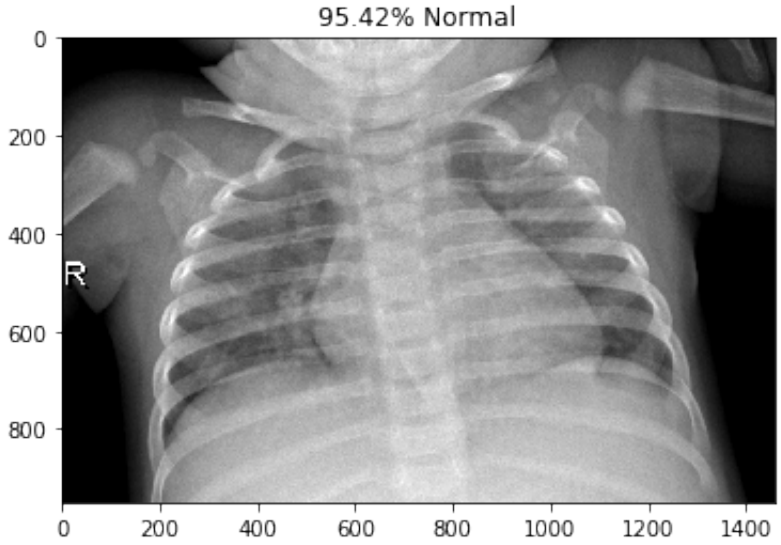
normal\NORMAL2-IM-1412-0001.jpeg



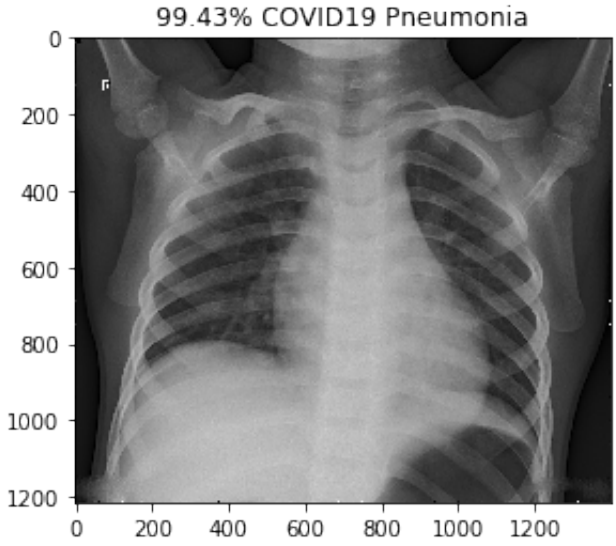
normal\NORMAL2-IM-1419-0001.jpeg



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.

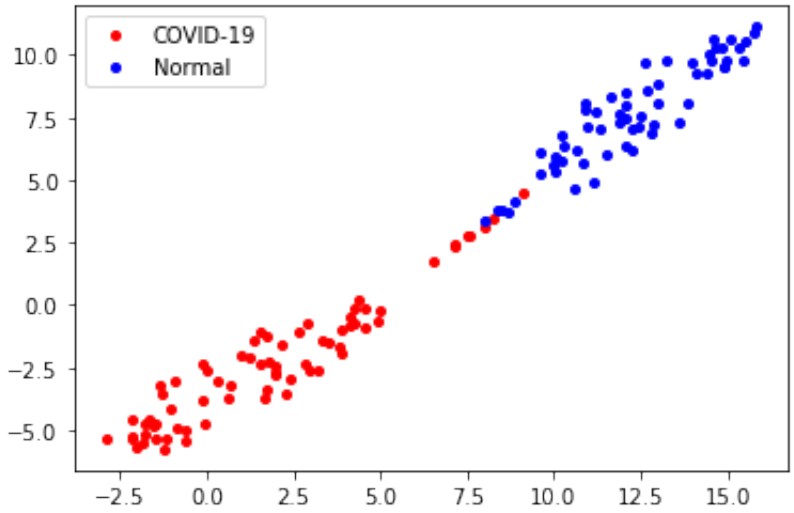
In [9]:

```
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.models.Model(inputs=model.input,
                                         outputs=model.get_layer('dense').output)
tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=
IMAGE_SIZE,
                                                       batch_size=1,shuffle=False,see
d=42,class_mode="binary")

output = intermediate_layer_model.predict_generator(tsne_data_generator,130,verb
ose=1)
feature = TSNE(n_components=2).fit_transform(output)
label = tsne_data_generator.classes

covid_x, covid_y, normal_x, normal_y = [],[],[],[]

plt.figure()

for index in range(len(feature)):
    if label[index] == 0:
        covid_x.append(feature[index, 0])
        covid_y.append(feature[index, 1])
    else:
        normal_x.append(feature[index, 0])
        normal_y.append(feature[index, 1])


plt.plot(covid_x, covid_y,'ro', ms=4, label="COVID-19")
plt.plot(normal_x, normal_y,'bo', ms=4, label="Normal")
plt.legend(loc='upper left')
```

```
Found 130 images belonging to 2 classes.
  1/130 [..............................] - ETA: 20s

C:\Users\Li\anaconda3\lib\site-packages\keras\engine\training.py:203
5: UserWarning: `Model.predict_generator` is deprecated and will be
removed in a future version. Please use `Model.predict`, which suppo
rts generators.
  warnings.warn('`Model.predict_generator` is deprecated and '

130/130 [==============================] - 8s 58ms/step
```

Out[9]:

```
<matplotlib.legend.Legend at 0x26465826308>
```



Bonus

In [10]:

```
print('With CPU: \n')

print('Devices:', tf.config.list_physical_devices())

print('Time:', total_time)
```

```
With CPU:

Devices: [PhysicalDevice(name='/physical_device:CPU:0', device_type=
'CPU')]
Time: 261.01105093955994
```

In [10]:

```
print('With CPU and GPU: \n')

print('Devices:', tf.config.list_physical_devices())

print('Time:', total_time)
```

```
With CPU and GPU:

Devices: [PhysicalDevice(name='/physical_device:CPU:0', device_type=
'CPU'), PhysicalDevice(name='/physical_device:GPU:0', device_type='G
PU')]
Time: 128.64543533325195
```