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)](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_VISIBLE_DEVICES'] = '-1'
tf.__version__
```

Out[1]:
```
'2.3.1'
```

Load Image Data

In [45]:
```
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
NUM_EPOCHS    = 40
LEARNING_RATE = 0.0005 # start off with high rate first 0.001 and expe
```

Generate Training and Validation Batches

In [4]:
```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,fe
                                   featurewise_std_normalization = Tru
                                   height_shift_range=0.2,shear_range=
                                   zca_whitening = True,channel_shift_
                                   horizontal_flip = True,vertical_fli
                                   validation_split = 0.2,fill_mode='c

train_batches = train_datagen.flow_from_directory(DATASET_PATH,target_
                                                  shuffle=True,batch_s
                                                  subset = "training",
                                                  class_mode="binary")

valid_batches = train_datagen.flow_from_directory(DATASET_PATH,target_
                                                  shuffle=True,batch_s
                                                  subset = "validation
                                                  class_mode="binary")
```

```
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.

In [10]:
```
VGG16 = tf.keras.applications.VGG16(weights="imagenet", include_top =

VGG16.trainable = False

model = tf.keras.Sequential([
    VGG16,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu', name='dense_feature'
    tf.keras.layers.Dense(1, activation='sigmoid'),
])

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               6422784
_________________________________________________________________
dense_2 (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 [11]:
```
#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

opt = tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE)
model.compile(optimizer=opt,
                loss=tf.keras.losses.BinaryCrossentropy(from_logits=Fa
                metrics=['accuracy'])

#need to specify step size
history = model.fit(train_batches, batch_size=BATCH_SIZE, steps_per_ep
                    epochs=NUM_EPOCHS, validation_data=valid_batches,
```

11

```
11
3
Epoch 1/40
10/10 [==============================] - 20s 2s/step - loss: 2.2660 - accuracy: 0.4362 - val_loss: 0.6367 - val_accuracy: 0.5500
Epoch 2/40
10/10 [==============================] - 20s 2s/step - loss: 0.5374 - accuracy: 0.7766 - val_loss: 0.4125 - val_accuracy: 1.0000
Epoch 3/40
10/10 [==============================] - 20s 2s/step - loss: 0.4139 - accuracy: 0.8191 - val_loss: 0.3330 - val_accuracy: 0.9000
Epoch 4/40
10/10 [==============================] - 22s 2s/step - loss: 0.3749 - accuracy: 0.8511 - val_loss: 0.3103 - val_accuracy: 0.9000
Epoch 5/40
10/10 [==============================] - 21s 2s/step - loss: 0.2895 - accuracy: 0.9149 - val_loss: 0.2533 - val_accuracy: 0.9500
Epoch 6/40
10/10 [==============================] - 20s 2s/step - loss: 0.2515 - accuracy: 0.9043 - val_loss: 0.1713 - val_accuracy: 1.0000
Epoch 7/40
10/10 [==============================] - 20s 2s/step - loss: 0.2754 - accuracy: 0.8723 - val_loss: 0.1716 - val_accuracy: 1.0000
Epoch 8/40
10/10 [==============================] - 20s 2s/step - loss: 0.2781 - accuracy: 0.8723 - val_loss: 0.1449 - val_accuracy: 0.9500
Epoch 9/40
10/10 [==============================] - 20s 2s/step - loss: 0.2175 - accuracy: 0.9149 - val_loss: 0.1440 - val_accuracy: 0.9500
Epoch 10/40
10/10 [==============================] - 22s 2s/step - loss: 0.2278 - accuracy: 0.8830 - val_loss: 0.1334 - val_accuracy: 0.9500
Epoch 11/40
10/10 [==============================] - 23s 2s/step - loss: 0.1709 - accuracy: 0.9362 - val_loss: 0.0704 - val_accuracy: 1.0000
Epoch 12/40
10/10 [==============================] - 21s 2s/step - loss: 0.1934 - accuracy: 0.9149 - val_loss: 0.0866 - val_accuracy: 1.0000
Epoch 13/40
10/10 [==============================] - 23s 2s/step - loss: 0.1950 - accuracy: 0.9255 - val_loss: 0.1082 - val_accuracy: 1.0000
Epoch 14/40
10/10 [==============================] - 27s 3s/step - loss: 0.2531 - accuracy: 0.8830 - val_loss: 0.0815 - val_accuracy: 1.0000
Epoch 15/40
10/10 [==============================] - 23s 2s/step - loss: 0.1935 - accuracy: 0.9149 - val_loss: 0.1076 - val_accuracy: 0.9500
Epoch 16/40
10/10 [==============================] - 22s 2s/step - loss: 0.1802 - accuracy: 0.9043 - val_loss: 0.0927 - val_accuracy: 1.0000
```

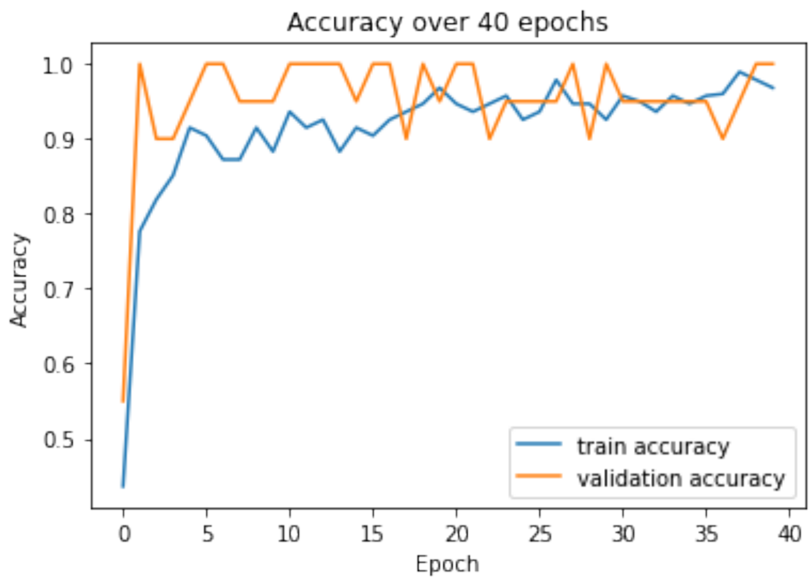
```
Epoch 17/40
10/10 [==============================] - 22s 2s/step - loss: 0.1539 -
accuracy: 0.9255 - val_loss: 0.0676 - val_accuracy: 1.0000
Epoch 18/40
10/10 [==============================] - 25s 3s/step - loss: 0.1734 -
accuracy: 0.9362 - val_loss: 0.1986 - val_accuracy: 0.9000
Epoch 19/40
10/10 [==============================] - 21s 2s/step - loss: 0.1547 -
accuracy: 0.9468 - val_loss: 0.0554 - val_accuracy: 1.0000
Epoch 20/40
10/10 [==============================] - 22s 2s/step - loss: 0.1115 -
accuracy: 0.9681 - val_loss: 0.1136 - val_accuracy: 0.9500
Epoch 21/40
10/10 [==============================] - 20s 2s/step - loss: 0.1682 -
accuracy: 0.9468 - val_loss: 0.0730 - val_accuracy: 1.0000
Epoch 22/40
10/10 [==============================] - 20s 2s/step - loss: 0.1338 -
accuracy: 0.9362 - val_loss: 0.0461 - val_accuracy: 1.0000
Epoch 23/40
10/10 [==============================] - 22s 2s/step - loss: 0.1389 -
accuracy: 0.9468 - val_loss: 0.1135 - val_accuracy: 0.9000
Epoch 24/40
10/10 [==============================] - 20s 2s/step - loss: 0.1290 -
accuracy: 0.9574 - val_loss: 0.0973 - val_accuracy: 0.9500
Epoch 25/40
10/10 [==============================] - 23s 2s/step - loss: 0.1687 -
accuracy: 0.9255 - val_loss: 0.1866 - val_accuracy: 0.9500
Epoch 26/40
10/10 [==============================] - 20s 2s/step - loss: 0.1538 -
accuracy: 0.9362 - val_loss: 0.0764 - val_accuracy: 0.9500
Epoch 27/40
10/10 [==============================] - 23s 2s/step - loss: 0.0924 -
accuracy: 0.9787 - val_loss: 0.1195 - val_accuracy: 0.9500
Epoch 28/40
10/10 [==============================] - 20s 2s/step - loss: 0.1219 -
accuracy: 0.9468 - val_loss: 0.0503 - val_accuracy: 1.0000
Epoch 29/40
10/10 [==============================] - 22s 2s/step - loss: 0.1790 -
accuracy: 0.9468 - val_loss: 0.1767 - val_accuracy: 0.9000
Epoch 30/40
10/10 [==============================] - 22s 2s/step - loss: 0.1504 -
accuracy: 0.9255 - val_loss: 0.0751 - val_accuracy: 1.0000
Epoch 31/40
10/10 [==============================] - 21s 2s/step - loss: 0.1098 -
accuracy: 0.9574 - val_loss: 0.0790 - val_accuracy: 0.9500
Epoch 32/40
10/10 [==============================] - 24s 2s/step - loss: 0.1051 -
accuracy: 0.9500 - val_loss: 0.0627 - val_accuracy: 0.9500
Epoch 33/40
10/10 [==============================] - 25s 2s/step - loss: 0.1137 -
```

```
accuracy: 0.9362 - val_loss: 0.1768 - val_accuracy: 0.9500
Epoch 34/40
10/10 [==============================] - 23s 2s/step - loss: 0.1159 -
accuracy: 0.9574 - val_loss: 0.0918 - val_accuracy: 0.9500
Epoch 35/40
10/10 [==============================] - 25s 3s/step - loss: 0.1692 -
accuracy: 0.9468 - val_loss: 0.0831 - val_accuracy: 0.9500
Epoch 36/40
10/10 [==============================] - 24s 2s/step - loss: 0.0772 -
accuracy: 0.9574 - val_loss: 0.1448 - val_accuracy: 0.9500
Epoch 37/40
10/10 [==============================] - 24s 2s/step - loss: 0.0969 -
accuracy: 0.9600 - val_loss: 0.1331 - val_accuracy: 0.9000
Epoch 38/40
10/10 [==============================] - 23s 2s/step - loss: 0.0743 -
accuracy: 0.9894 - val_loss: 0.1506 - val_accuracy: 0.9500
Epoch 39/40
10/10 [==============================] - 22s 2s/step - loss: 0.0954 -
accuracy: 0.9787 - val_loss: 0.0474 - val_accuracy: 1.0000
Epoch 40/40
10/10 [==============================] - 24s 2s/step - loss: 0.0828 -
accuracy: 0.9681 - val_loss: 0.0704 - val_accuracy: 1.0000
```

[5 points] Plot Accuracy and Loss During Training

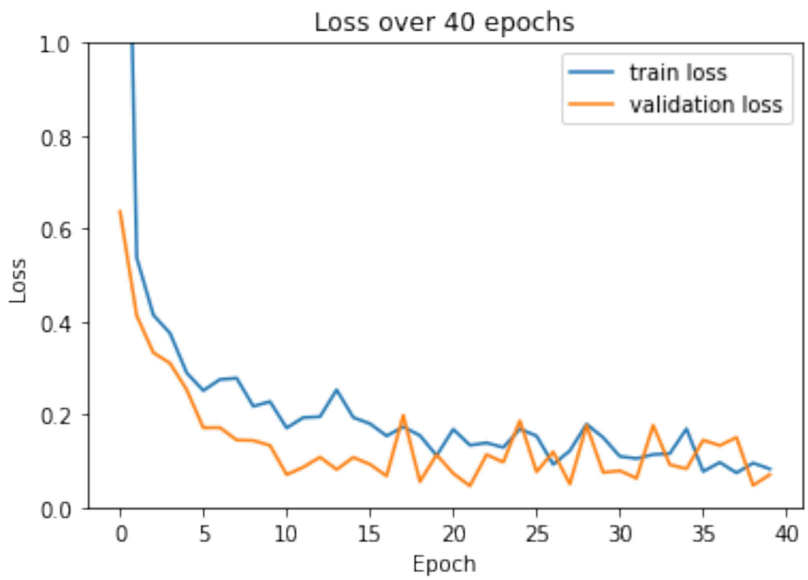
In [13]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label = 'validation accuracy
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.title('Accuracy over ' + str(NUM_EPOCHS) + ' epochs')
```

Out[13]:
```
Text(0.5, 1.0, 'Accuracy over 40 epochs')
```



In [81]:
```
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label = 'validation loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.ylim([0, 1])
plt.legend(loc='upper right')
plt.title('Loss over ' + str(NUM_EPOCHS) + ' epochs')
#print(str(history.history['loss']))
```

Out[81]:
```
Text(0.5, 1.0, 'Loss over 40 epochs')
```

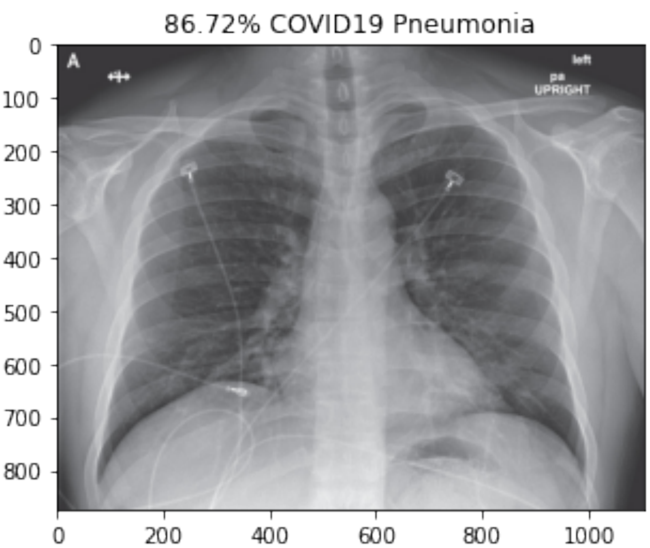


Plot Test Results

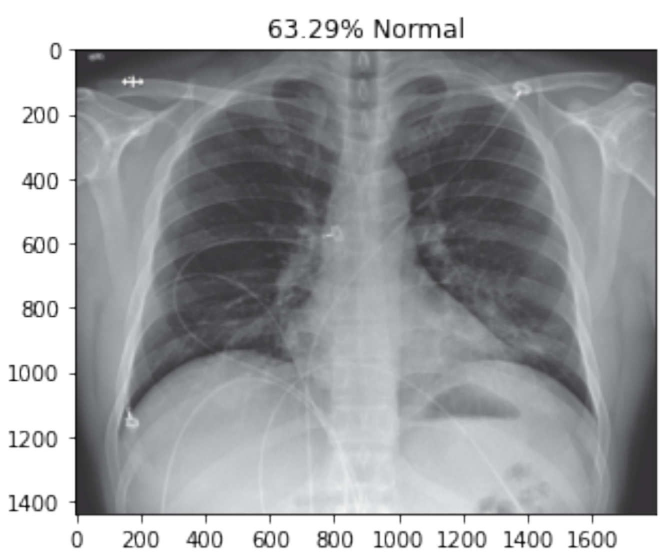
In [17]:
```
import matplotlib.image as mpimg

test_datagen = ImageDataGenerator(rescale=1. / 255)
eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size
                                                  batch_size=1,shuffle
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 Pneum
    plt.show()
```

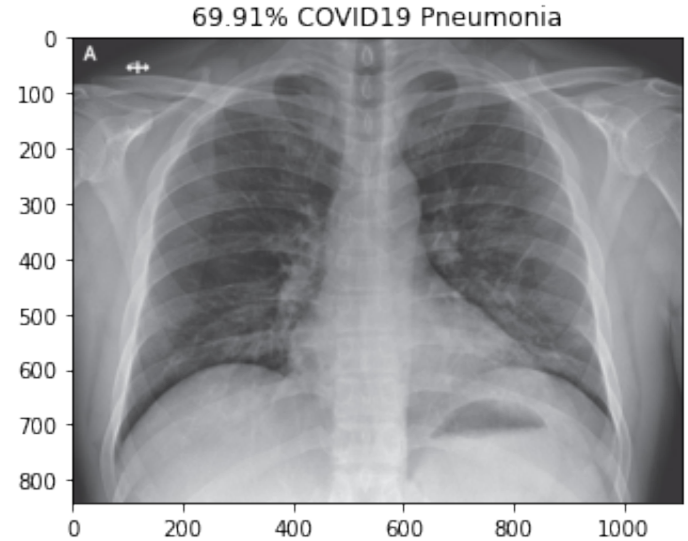
```
Found 18 images belonging to 2 classes.
WARNING:tensorflow:From <ipython-input-17-543347a5fba8>:7: Model.pred
ict_generator (from tensorflow.python.keras.engine.training) is depre
cated and will be removed in a future version.
Instructions for updating:
Please use Model.predict, which supports generators.
18/18 [==============================] - 4s 203ms/step
covid/nejmoa2001191_f3-PA.jpeg
```



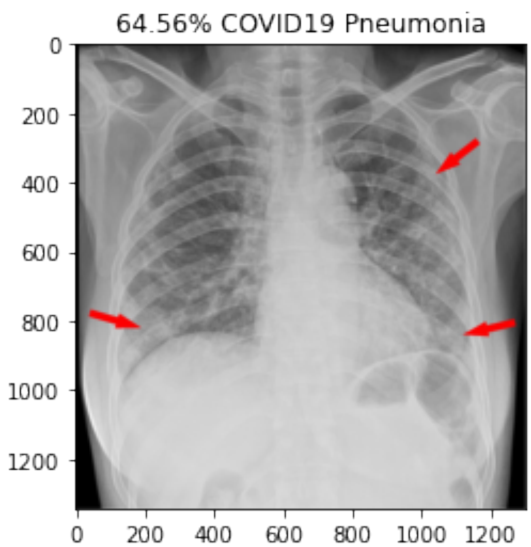

```
covid/nejmoa2001191_f4.jpeg
```



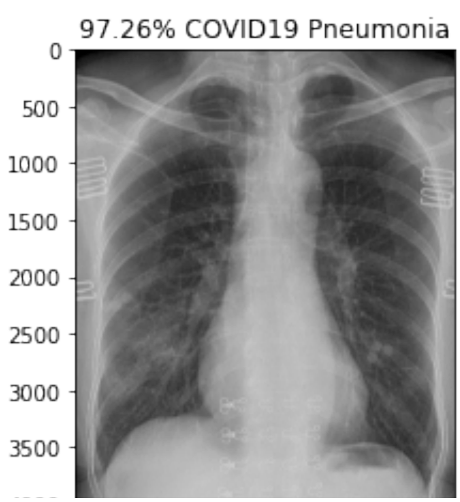

```
covid/nejmoa2001191_f5-PA.jpeg
```

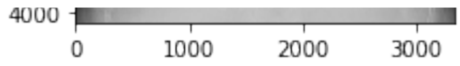


covid/radiol.2020200490.fig3.jpeg

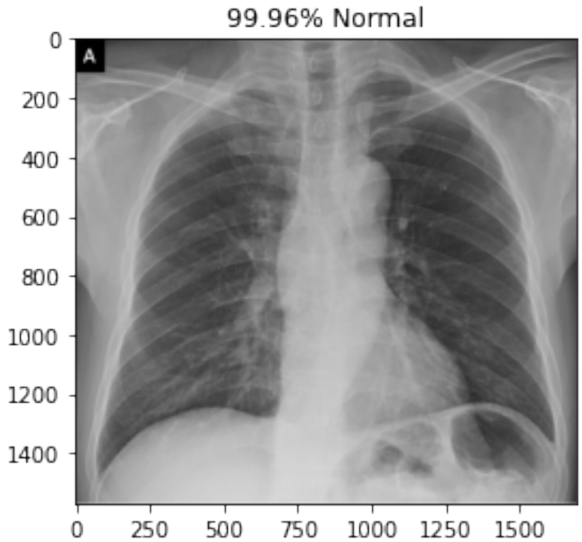


covid/ryct.2020200028.fig1a.jpeg

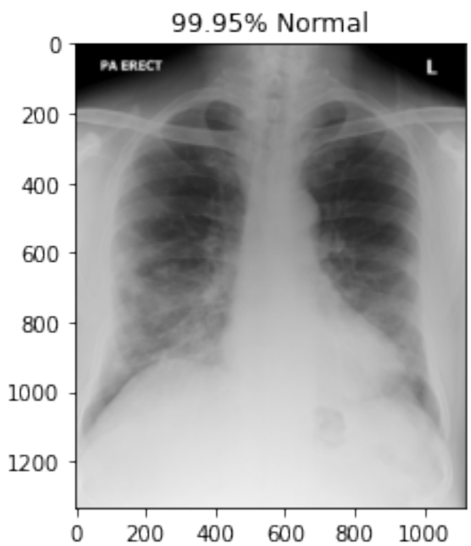




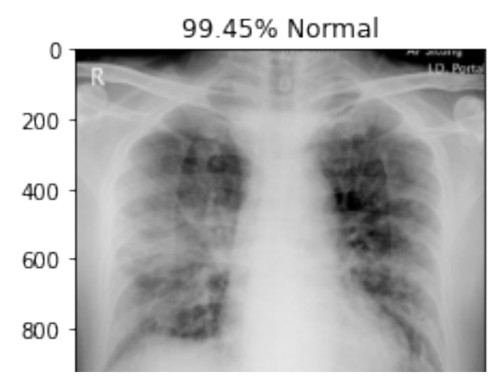
covid/ryct.2020200034.fig2.jpeg

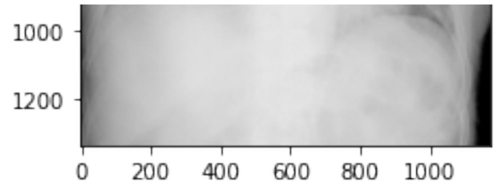


covid/ryct.2020200034.fig5-day0.jpeg

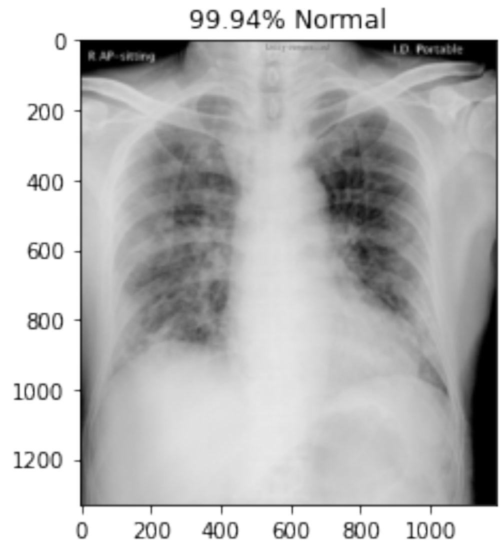


covid/ryct.2020200034.fig5-day4.jpeg

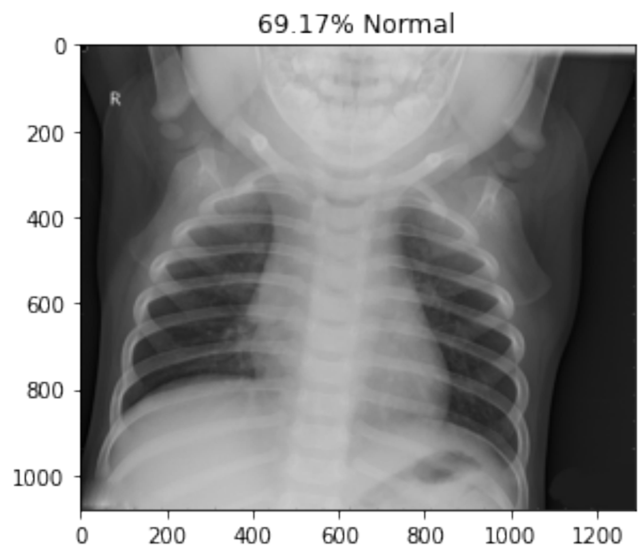




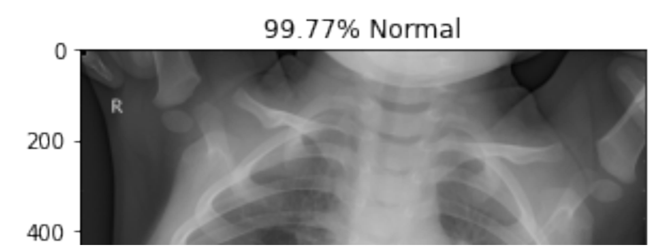
covid/ryct.2020200034.fig5-day7.jpeg

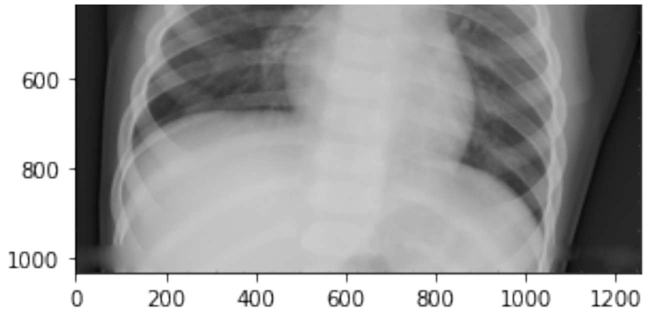


normal/NORMAL2-IM-1385-0001.jpeg

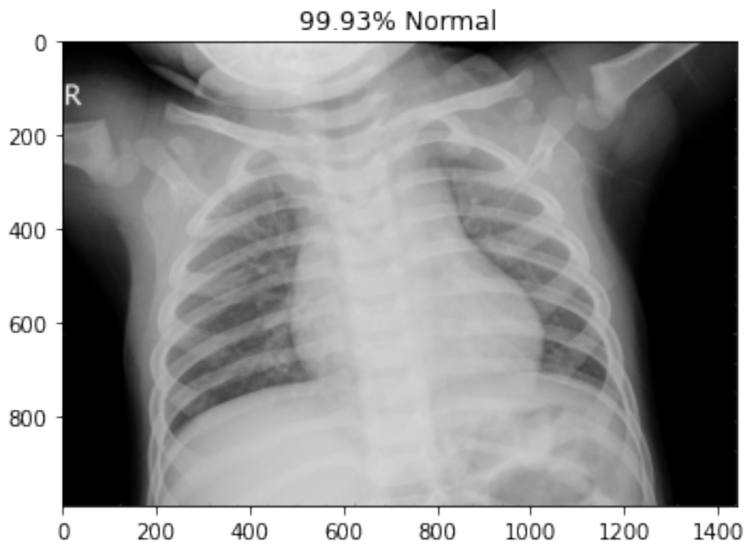


normal/NORMAL2-IM-1396-0001.jpeg

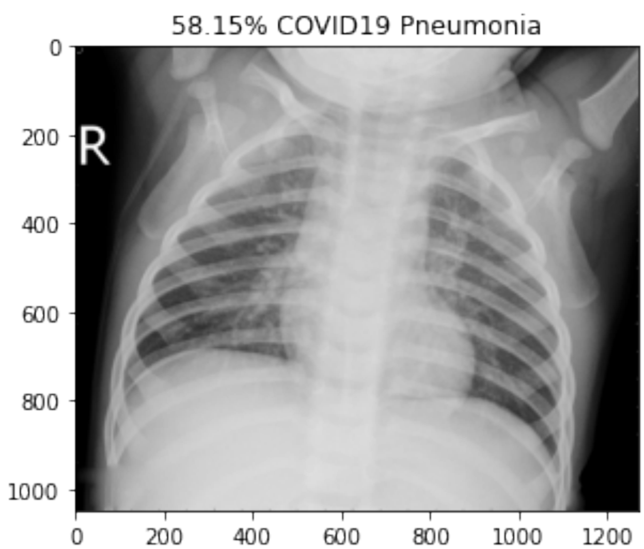




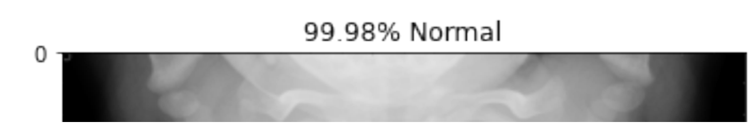

```
normal/NORMAL2-IM-1400-0001.jpeg
```

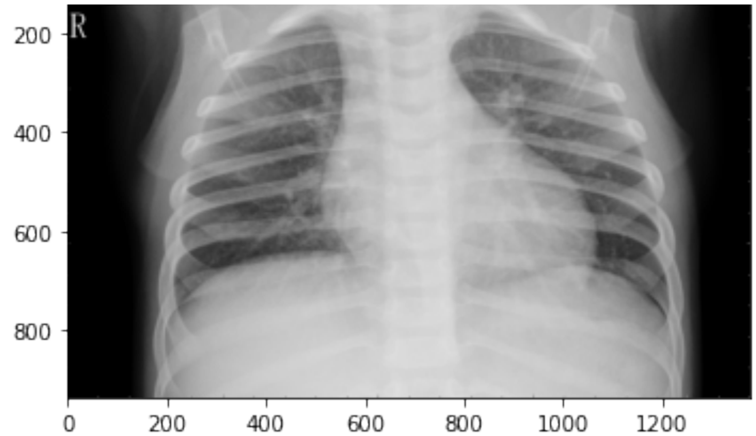



```
normal/NORMAL2-IM-1401-0001.jpeg
```

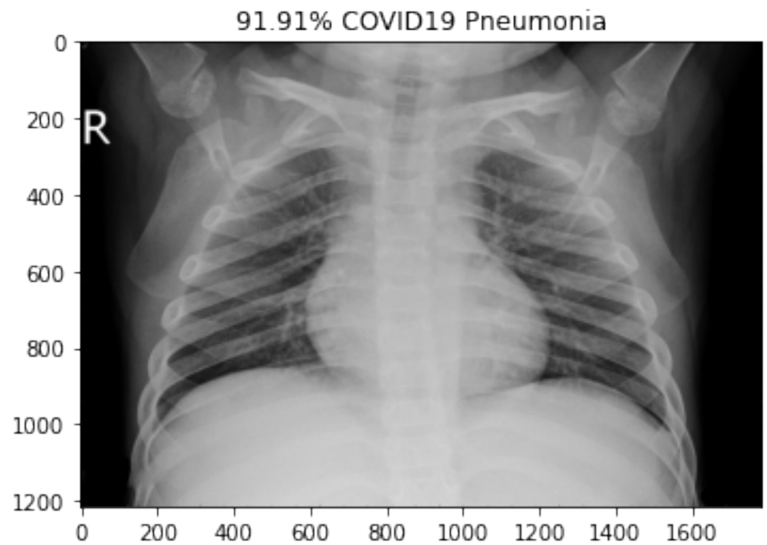



```
normal/NORMAL2-IM-1406-0001.jpeg
```

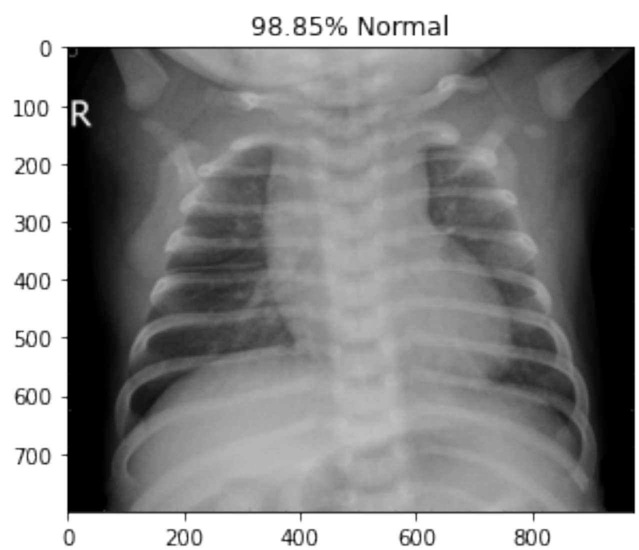




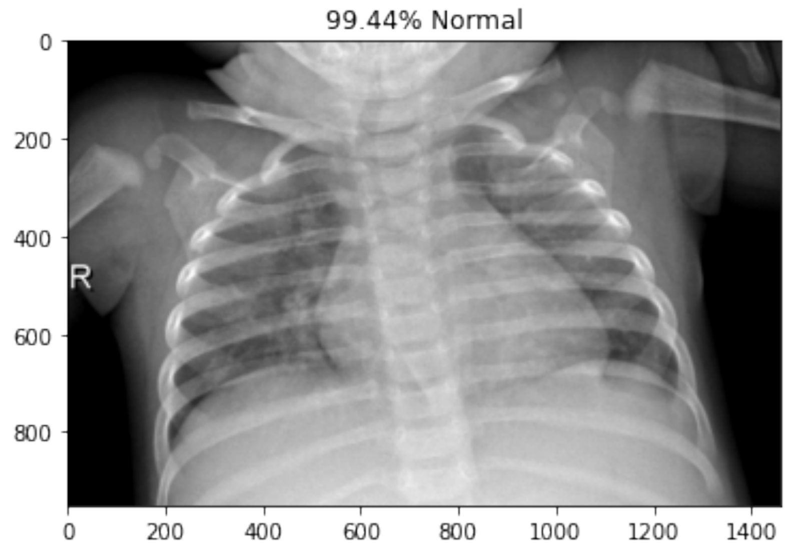
normal/NORMAL2-IM-1412-0001.jpeg



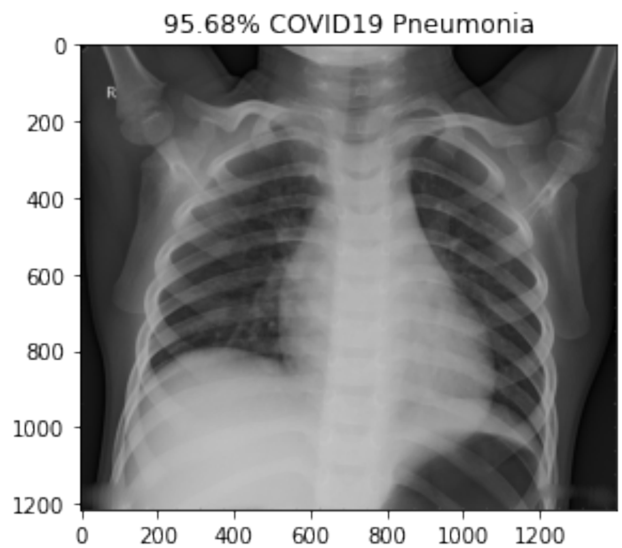
normal/NORMAL2-IM-1419-0001.jpeg



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 [89]:
```
from sklearn.manifold import TSNE
from tensorflow.keras import models

intermediate_layer_model = models.Model(inputs=model.input,
                                        outputs=model.get_layer('dense
```

```
tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,ta
                                                       batch_size=1,shuffle

# print(train_batches.labels)

# print(valid_batches.labels)

# print((len(train_batches.labels) + len(valid_batches.labels)))

# print(tsne_data_generator.labels)

#print(str(model.output[0]))
# print("intermediate layer model: ")
# print(intermediate_layer_model.input.shape)
# print(intermediate_layer_model.output.shape)

layer = intermediate_layer_model.predict(tsne_data_generator)

# (COMMENTED OUT)
# print(pred.shape)
# print(layer.shape)
# print(tsne_data_generator[1])
# print(pred)
# print(str(layer[0][0]))
# print(len(layer))

labels = tsne_data_generator.labels
tsne = TSNE(n_components=2)
intermediate_tsne = tsne.fit_transform(layer)

# print("Len of TSNE: " + str(len(intermediate_tsne)))

color_array = []
for i in range(len(labels)):
    if labels[i] > 0.5:
        color_array += 'b'
    else:
        color_array += 'r'

# print(pred)
# print(color_array)
print("\nLEGEND:")
print("blue = Normal")
print("red = Covid")

plt.figure(figsize=(8, 8))
plt.scatter(x = intermediate_tsne[:,0], y=intermediate_tsne[:,1], c=co
plt.show()
# print("OK")
```

Found 130 images belonging to 2 classes

```
Found 150 images belonging to 2 classes.

LEGEND:
blue = Normal
red = Covid
```