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

In [10]:

```
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[10]:

```
'2.4.1'
```

Load Image Data

In [11]:

```
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 = 0.001 # start off with high rate first 0.001 and experiment with reducing it gradually
```

Generate Training and Validation Batches

In [12]:

```
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.
```

[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 [13]:

```
# raise NotImplementedError("Build your model based on an architecture of your choice "
#                           "A sample model summary is shown below")
from tensorflow.keras.applications.vgg16 import VGG16

model = tf.keras.Sequential()
vgg16 = VGG16(input_shape=(224,224,3),include_top=False, weights='imagenet')

# Make vgg16 untrainable as per specification
vgg16.trainable=False
model.add(vgg16)
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256,name='dense_feature',activation='relu'))
model.add(tf.keras.layers.Dense(1,activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_feature (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
_________________________________________________________________
```

REPORT

I used vgg16 pretrained model for this task, and reporduce the model summary provided with the notebook. No dropout layer was needed

I managed to recreate the model and obtained good results, therefore I did not change optimizer compared to default settings. At first I had softmax as last layer's activation method and obtained terrible results, changing it to sigmoid solved the problem. I think it is because the default setting for optimizer is from_logits=True, which does not work with softmax.

As far as loss function, I chose binary_crossentropy for binary classification, for obvious reasons.

[5 points] Train Model

In [14]:

```
#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(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
history = model.fit(
    train_batches,
    batch_size=BATCH_SIZE,
    epochs=NUM_EPOCHS,
    validation_data=valid_batches
)
#raise NotImplementedError("Use the model.fit function to train your network")
```

```
11
3
```

```
Epoch 1/40
11/11 [==============================] - 6s 561ms/step - loss: 2.4950 - accuracy: 0.5831 - val_loss: 0.7882 - val_ac
curacy: 0.7308
Epoch 2/40
11/11 [==============================] - 6s 520ms/step - loss: 0.7998 - accuracy: 0.6867 - val_loss: 0.0735 - val_ac
curacy: 1.0000
Epoch 3/40
11/11 [==============================] - 6s 522ms/step - loss: 0.2633 - accuracy: 0.9282 - val_loss: 0.1004 - val_ac
curacy: 0.9615
Epoch 4/40
11/11 [==============================] - 6s 521ms/step - loss: 0.3694 - accuracy: 0.9100 - val_loss: 0.3377 - val_ac
curacy: 0.8462
Epoch 5/40
11/11 [==============================] - 6s 545ms/step - loss: 0.3644 - accuracy: 0.8552 - val_loss: 0.1498 - val_ac
curacy: 0.9231
Epoch 6/40
11/11 [==============================] - 6s 545ms/step - loss: 0.3223 - accuracy: 0.9075 - val_loss: 0.1661 - val_ac
curacy: 0.9231
Epoch 7/40
11/11 [==============================] - 6s 524ms/step - loss: 0.1832 - accuracy: 0.9361 - val_loss: 0.1062 - val_ac
curacy: 0.9615
Epoch 8/40
11/11 [==============================] - 6s 532ms/step - loss: 0.1624 - accuracy: 0.9399 - val_loss: 0.0716 - val_ac
curacy: 0.9615
Epoch 9/40
11/11 [==============================] - 6s 511ms/step - loss: 0.1106 - accuracy: 0.9781 - val_loss: 0.3264 - val_ac
curacy: 0.8846
Epoch 10/40
11/11 [==============================] - 6s 524ms/step - loss: 0.3122 - accuracy: 0.9122 - val_loss: 0.1678 - val_ac
curacy: 0.9231
Epoch 11/40
11/11 [==============================] - 6s 513ms/step - loss: 0.2031 - accuracy: 0.9228 - val_loss: 0.0552 - val_ac
curacy: 1.0000
Epoch 12/40
11/11 [==============================] - 6s 520ms/step - loss: 0.1564 - accuracy: 0.9472 - val_loss: 0.1044 - val_ac
curacy: 0.9615
Epoch 13/40
11/11 [==============================] - 6s 538ms/step - loss: 0.1840 - accuracy: 0.9223 - val_loss: 0.1231 - val_ac
curacy: 0.9615
Epoch 14/40
11/11 [==============================] - 6s 514ms/step - loss: 0.1509 - accuracy: 0.9270 - val_loss: 0.0583 - val_ac
curacy: 1.0000
Epoch 15/40
11/11 [==============================] - 6s 513ms/step - loss: 0.1598 - accuracy: 0.9715 - val_loss: 0.1049 - val_ac
curacy: 0.9231
Epoch 16/40
11/11 [==============================] - 6s 524ms/step - loss: 0.0790 - accuracy: 0.9756 - val_loss: 0.0516 - val_ac
curacy: 1.0000
Epoch 17/40
11/11 [==============================] - 6s 514ms/step - loss: 0.0731 - accuracy: 0.9938 - val_loss: 0.2320 - val_ac
curacy: 0.9231
Epoch 18/40
11/11 [==============================] - 6s 520ms/step - loss: 0.1344 - accuracy: 0.9631 - val_loss: 0.0813 - val_ac
curacy: 1.0000
Epoch 19/40
11/11 [==============================] - 6s 511ms/step - loss: 0.0666 - accuracy: 0.9738 - val_loss: 0.1220 - val_ac
curacy: 0.9231
Epoch 20/40
11/11 [==============================] - 6s 513ms/step - loss: 0.2073 - accuracy: 0.9510 - val_loss: 0.0398 - val_ac
curacy: 1.0000
Epoch 21/40
11/11 [==============================] - 6s 512ms/step - loss: 0.0703 - accuracy: 0.9926 - val_loss: 0.0185 - val_ac
curacy: 1.0000
Epoch 22/40
11/11 [==============================] - 6s 513ms/step - loss: 0.0669 - accuracy: 0.9884 - val_loss: 0.1908 - val_ac
curacy: 0.9231
Epoch 23/40
11/11 [==============================] - 6s 511ms/step - loss: 0.0565 - accuracy: 0.9725 - val_loss: 0.1008 - val_ac
curacy: 0.9231
Epoch 24/40
11/11 [==============================] - 6s 514ms/step - loss: 0.0848 - accuracy: 0.9830 - val_loss: 0.0621 - val_ac
curacy: 0.9615
Epoch 25/40
11/11 [==============================] - 6s 516ms/step - loss: 0.1399 - accuracy: 0.9609 - val_loss: 0.1023 - val_ac
curacy: 0.9615
Epoch 26/40
11/11 [==============================] - 6s 520ms/step - loss: 0.0640 - accuracy: 0.9813 - val_loss: 0.1734 - val_ac
curacy: 0.9231
Epoch 27/40
11/11 [==============================] - 6s 535ms/step - loss: 0.0662 - accuracy: 0.9882 - val_loss: 0.1170 - val_ac
curacy: 0.9615
Epoch 28/40
11/11 [==============================] - 6s 514ms/step - loss: 0.1144 - accuracy: 0.9536 - val_loss: 0.1068 - val_ac
curacy: 0.9615
Epoch 29/40
11/11 [==============================] - 6s 536ms/step - loss: 0.4539 - accuracy: 0.8448 - val_loss: 0.1854 - val_ac
curacy: 0.9231
Epoch 30/40
11/11 [==============================] - 6s 510ms/step - loss: 0.0760 - accuracy: 0.9785 - val_loss: 0.0880 - val_ac
curacy: 0.9231
Epoch 31/40
11/11 [==============================] - 6s 517ms/step - loss: 0.0648 - accuracy: 0.9801 - val_loss: 0.1348 - val_ac
curacy: 0.9231
```

```
Epoch 32/40
11/11 [==============================] - 6s 514ms/step - loss: 0.1678 - accuracy: 0.9538 - val_loss: 0.4346 - val_ac
curacy: 0.8462
Epoch 33/40
11/11 [==============================] - 6s 513ms/step - loss: 0.0835 - accuracy: 0.9682 - val_loss: 0.0678 - val_ac
curacy: 0.9615
Epoch 34/40
11/11 [==============================] - 6s 511ms/step - loss: 0.0586 - accuracy: 0.9775 - val_loss: 0.0325 - val_ac
curacy: 1.0000
Epoch 35/40
11/11 [==============================] - 6s 510ms/step - loss: 0.0828 - accuracy: 0.9666 - val_loss: 0.2631 - val_ac
curacy: 0.9615
Epoch 36/40
11/11 [==============================] - 6s 513ms/step - loss: 0.1044 - accuracy: 0.9350 - val_loss: 0.0304 - val_ac
curacy: 1.0000
Epoch 37/40
11/11 [==============================] - 6s 520ms/step - loss: 0.1762 - accuracy: 0.9607 - val_loss: 0.2751 - val_ac
curacy: 0.8846
Epoch 38/40
11/11 [==============================] - 6s 515ms/step - loss: 0.0717 - accuracy: 0.9812 - val_loss: 0.3005 - val_ac
curacy: 0.8462
Epoch 39/40
11/11 [==============================] - 6s 532ms/step - loss: 0.0957 - accuracy: 0.9708 - val_loss: 0.2880 - val_ac
curacy: 0.9231
Epoch 40/40
11/11 [==============================] - 6s 521ms/step - loss: 0.3416 - accuracy: 0.8640 - val_loss: 0.1664 - val_ac
```

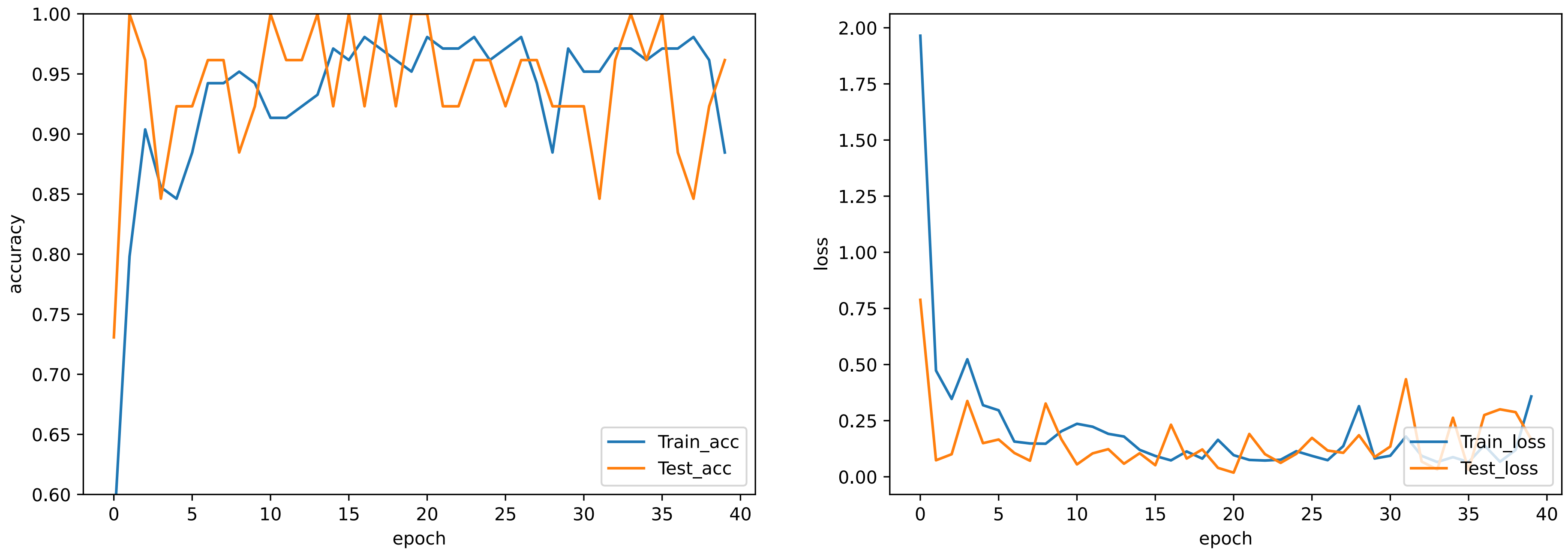
[5 points] Plot Accuracy and Loss During Training

In [15]:

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'],label='Train_acc')
plt.plot(history.history['val_accuracy'],label='Test_acc')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.ylim([0.6,1])
plt.legend(loc='lower right')

plt.subplot(1,2,2)
plt.plot(history.history['loss'],label='Train_loss')
plt.plot(history.history['val_loss'],label='Test_loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(loc='lower right')
# raise NotImplementedError("Plot the accuracy and the loss during training")
```

Out[15]:

```
<matplotlib.legend.Legend at 0x7fb1d86b2d60>
```

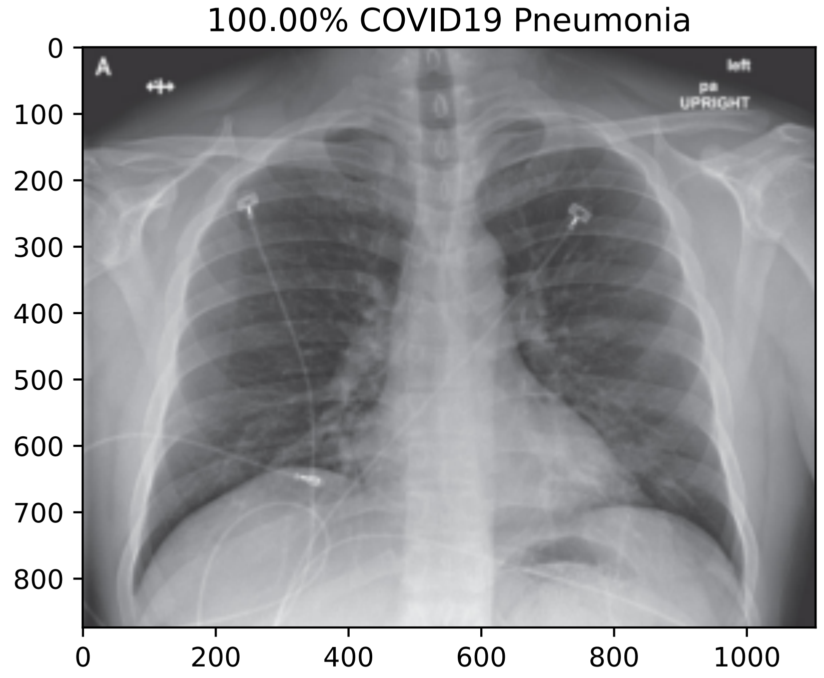


Plot Test Results

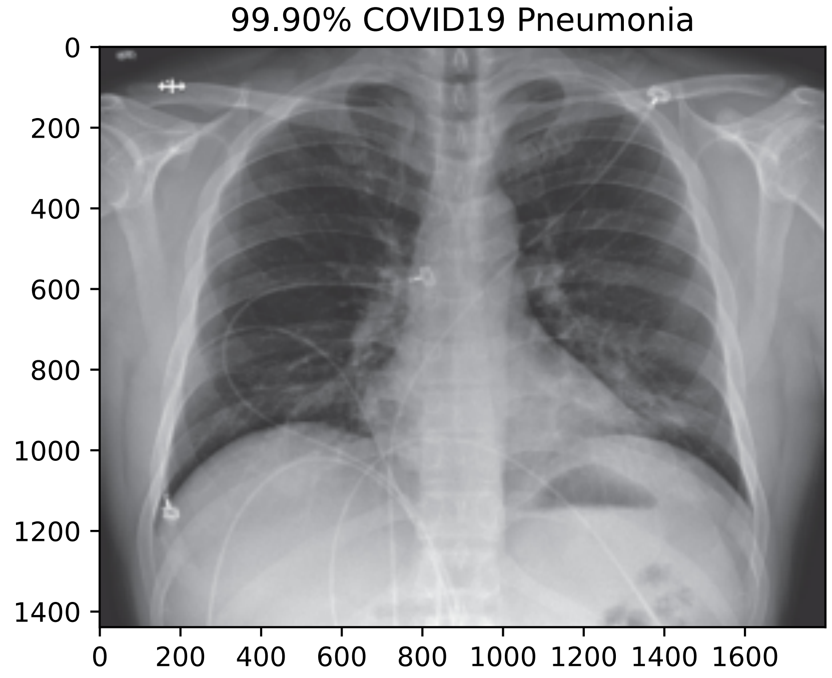
In [16]:

```
import matplotlib.image as mpimg

test_datagen = ImageDataGenerator(rescale=1. / 255)
eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                  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()
```

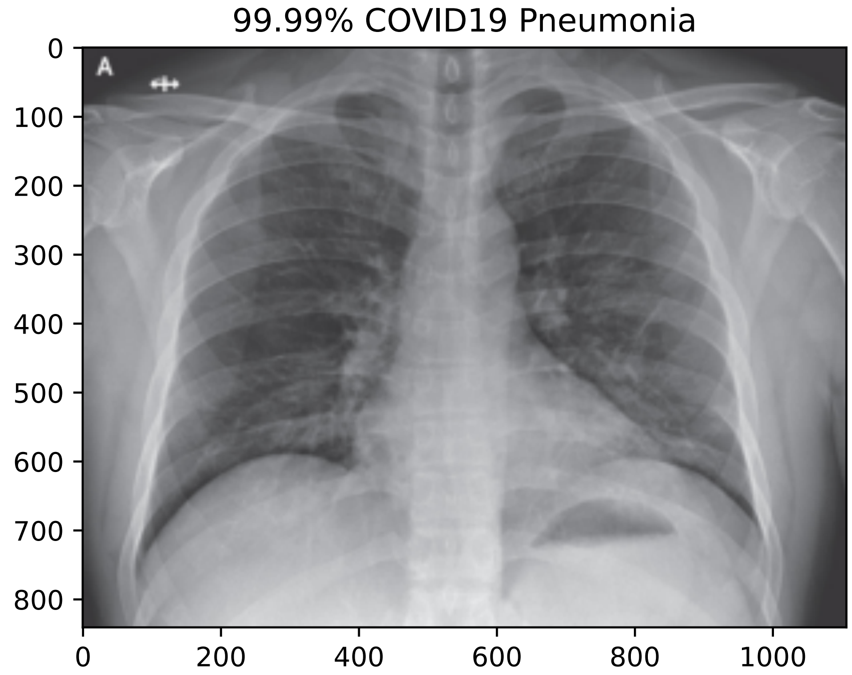
```
Found 18 images belonging to 2 classes.
 1/18 [>.............................] - ETA: 2s/home/shawn/.local/lib/python3.8/site-packages/tensorflow/python/ker
as/engine/training.py:1905: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future ver
sion. Please use `Model.predict`, which supports generators.
  warnings.warn('`Model.predict_generator` is deprecated and '
18/18 [==============================] - 1s 48ms/step
covid/nejmoa2001191_f3-PA.jpeg
```



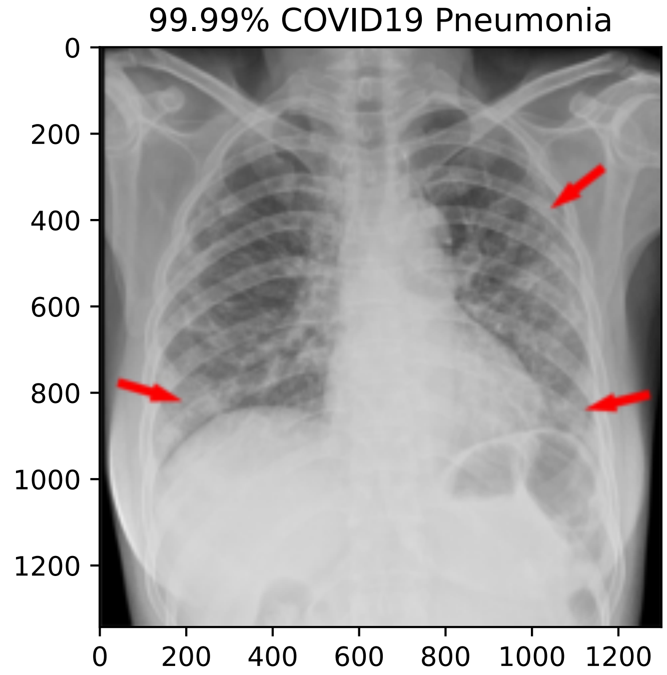
```
covid/nejmoa2001191_f4.jpeg
```



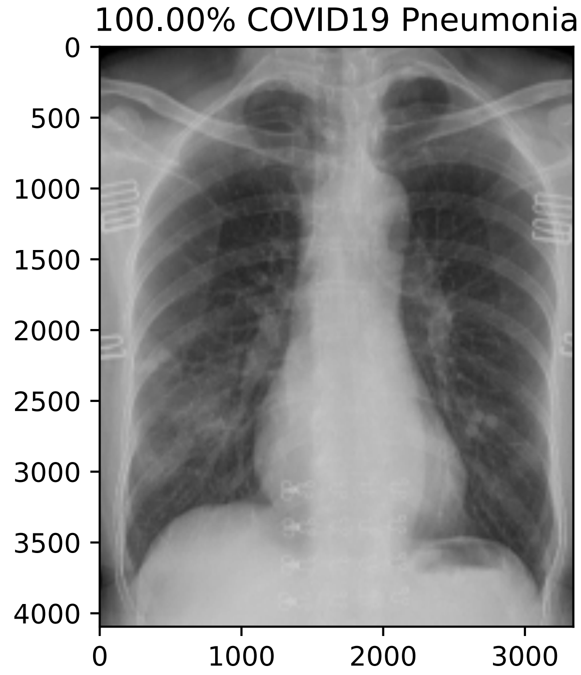
```
covid/nejmoa2001191_f5-PA.jpeg
```



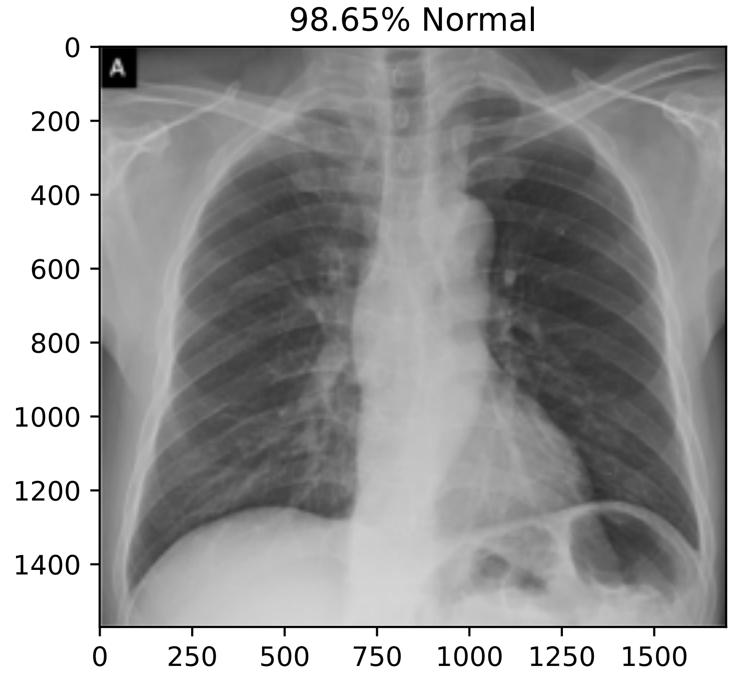
```
covid/radiol.2020200490.fig3.jpeg
```



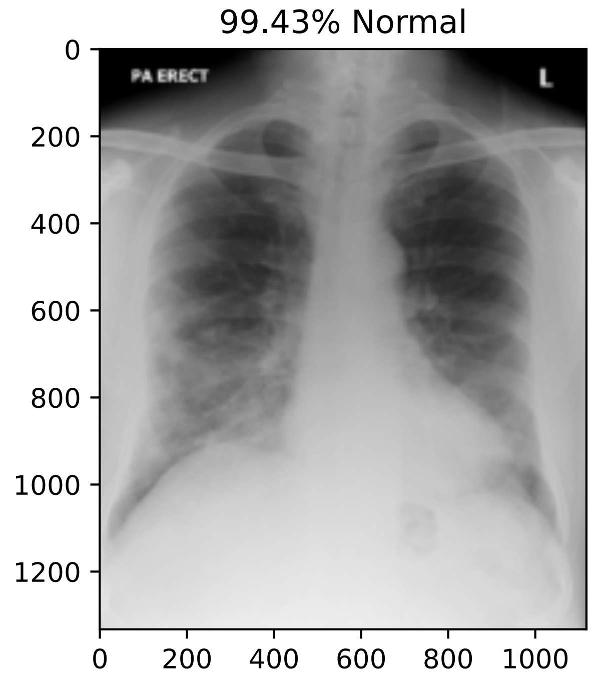
```
covid/ryct.2020200028.fig1a.jpeg
```



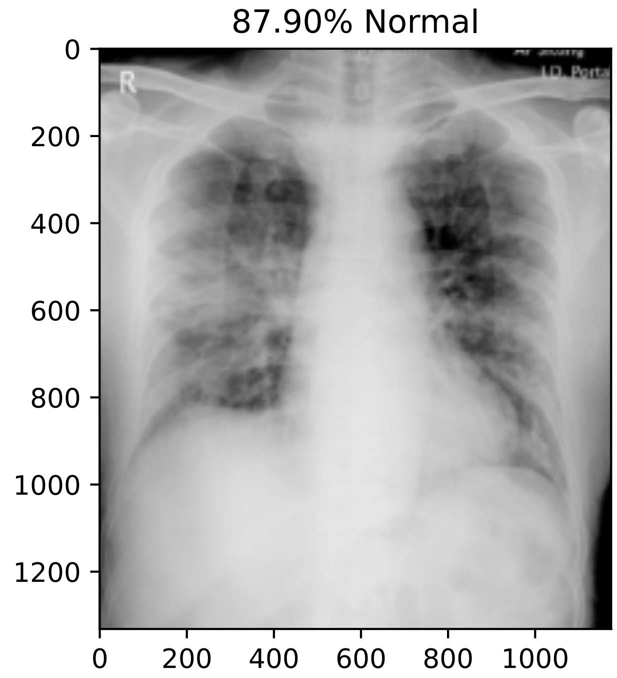
```
covid/ryct.2020200034.fig2.jpeg
```



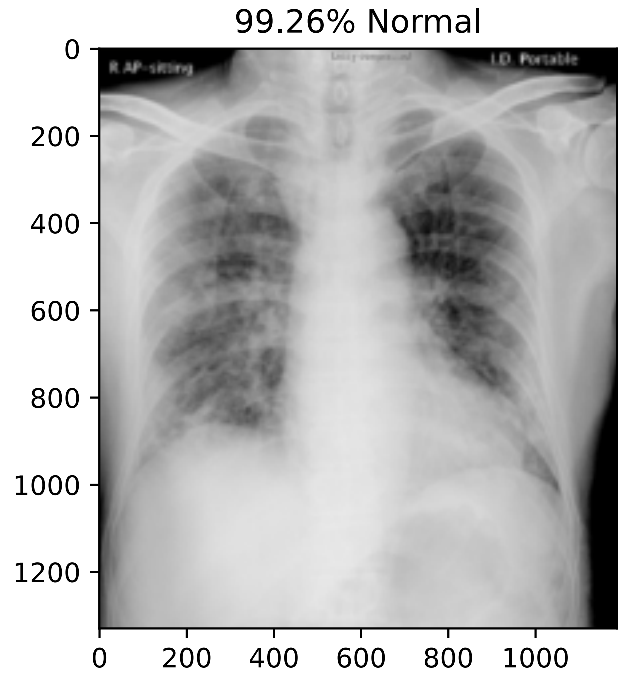
```
covid/ryct.2020200034.fig5-day0.jpeg
```



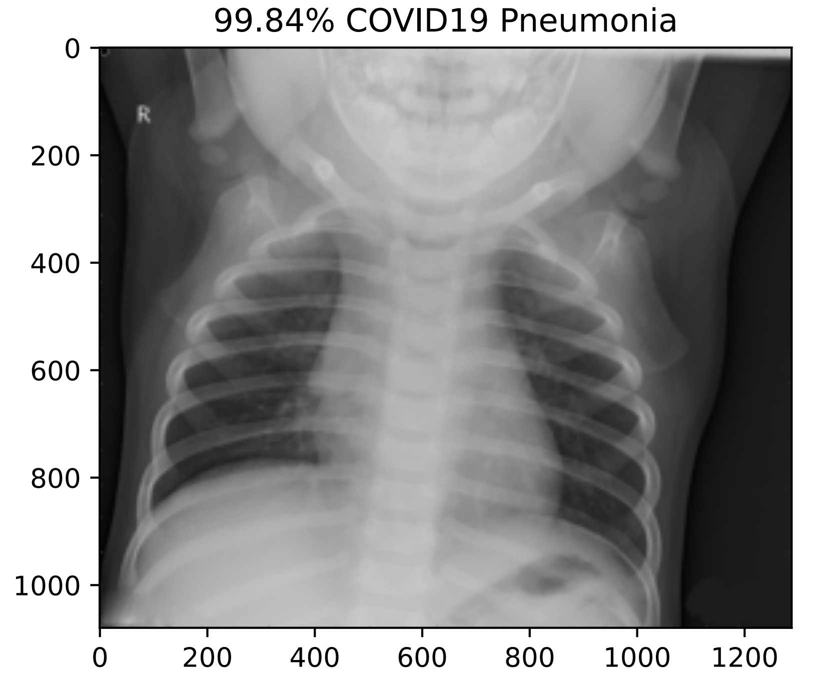
```
covid/ryct.2020200034.fig5-day4.jpeg
```



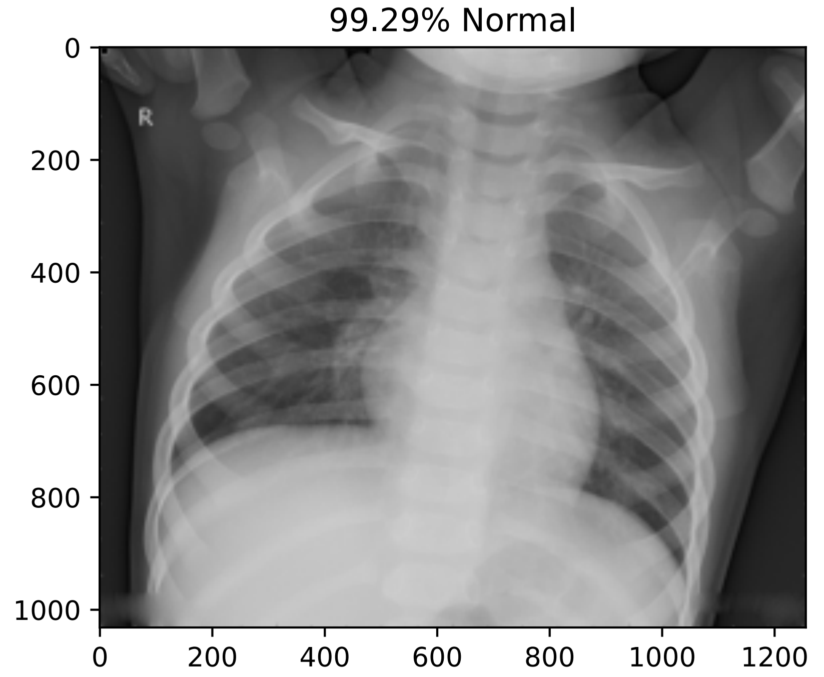
```
covid/ryct.2020200034.fig5-day7.jpeg
```



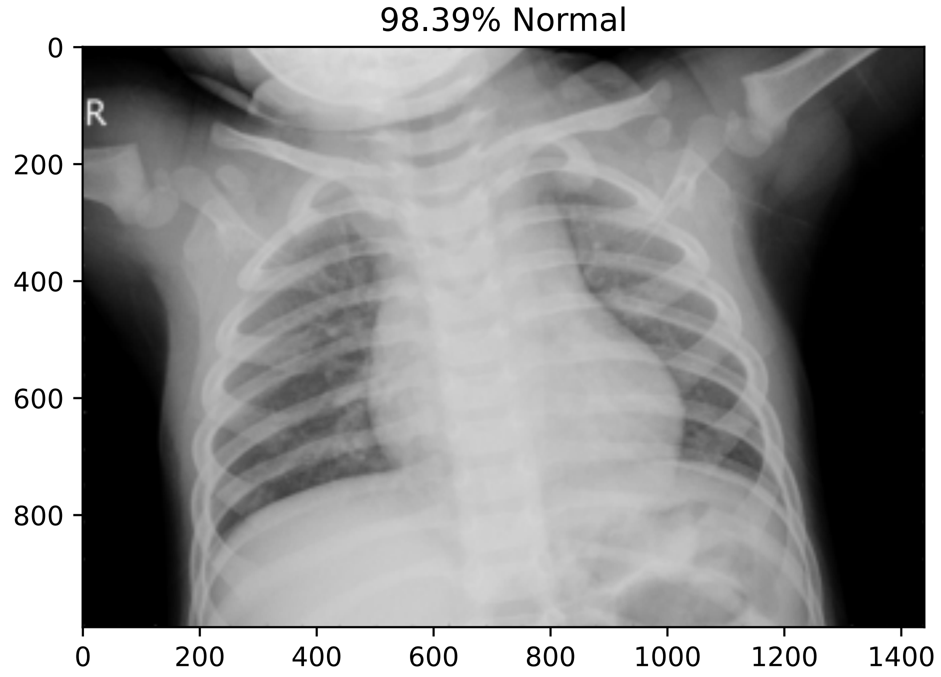
```
normal/NORMAL2-IM-1385-0001.jpeg
```



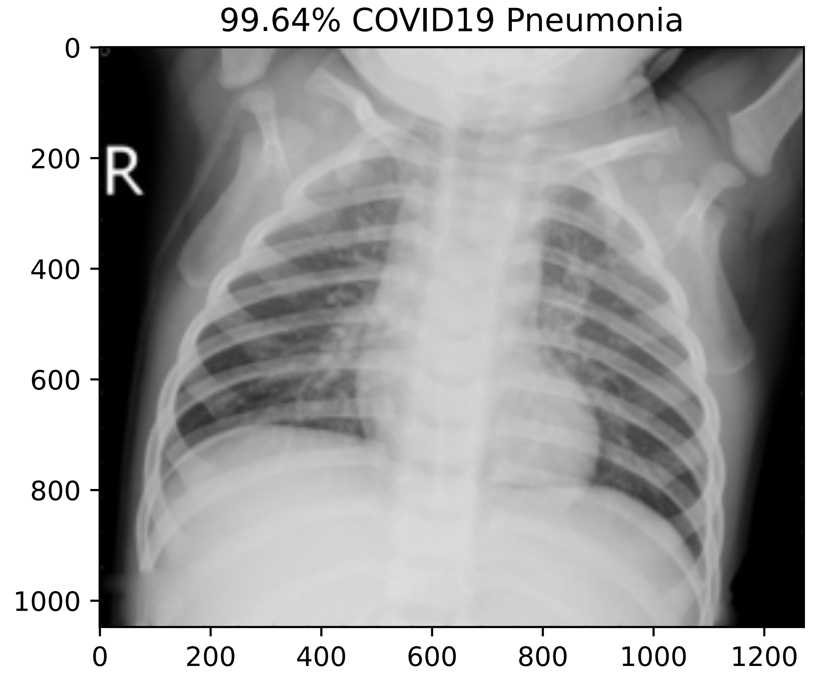
```
normal/NORMAL2-IM-1396-0001.jpeg
```



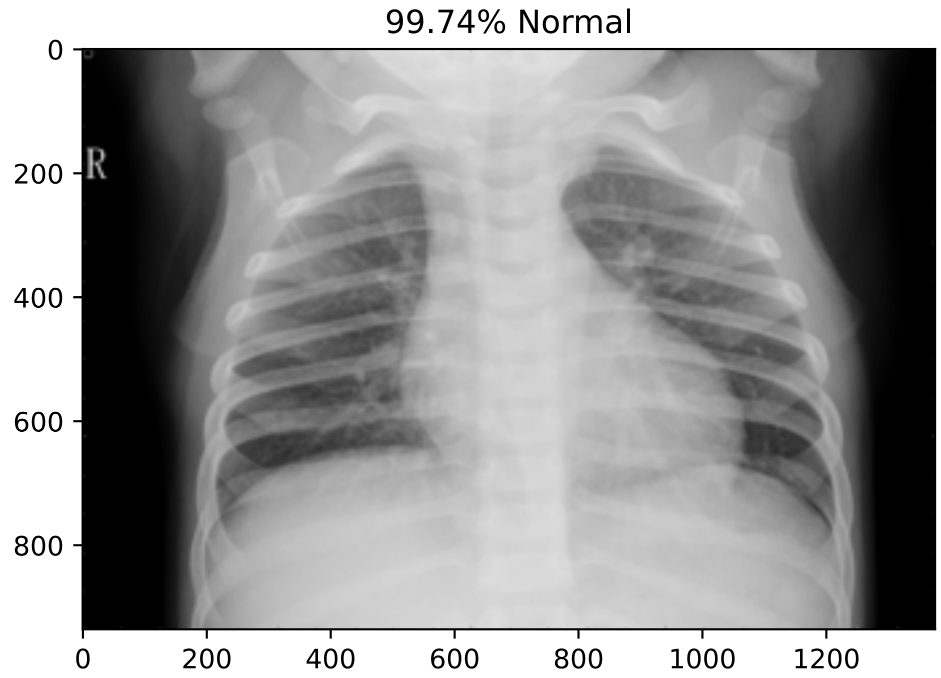
```
normal/NORMAL2-IM-1400-0001.jpeg
```



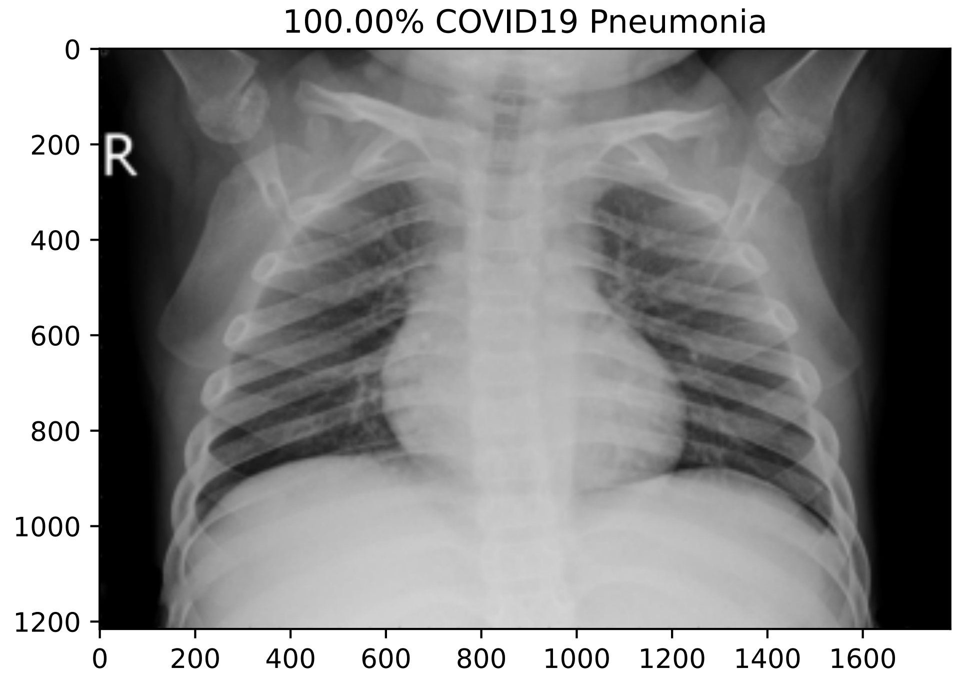
```
normal/NORMAL2-IM-1401-0001.jpeg
```



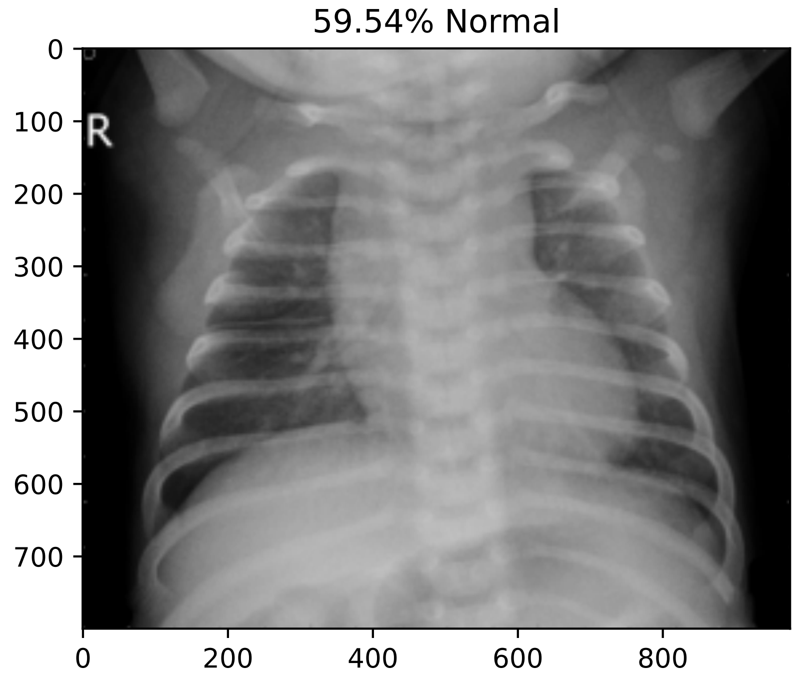
```
normal/NORMAL2-IM-1406-0001.jpeg
```



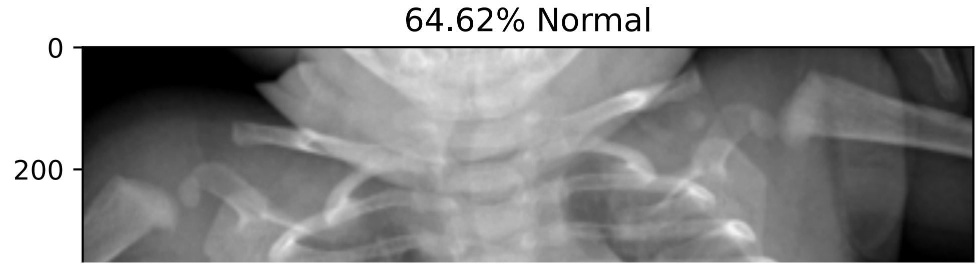
```
normal/NORMAL2-IM-1412-0001.jpeg
```



```
normal/NORMAL2-IM-1419-0001.jpeg
```



```
normal/NORMAL2-IM-1422-0001.jpeg
```



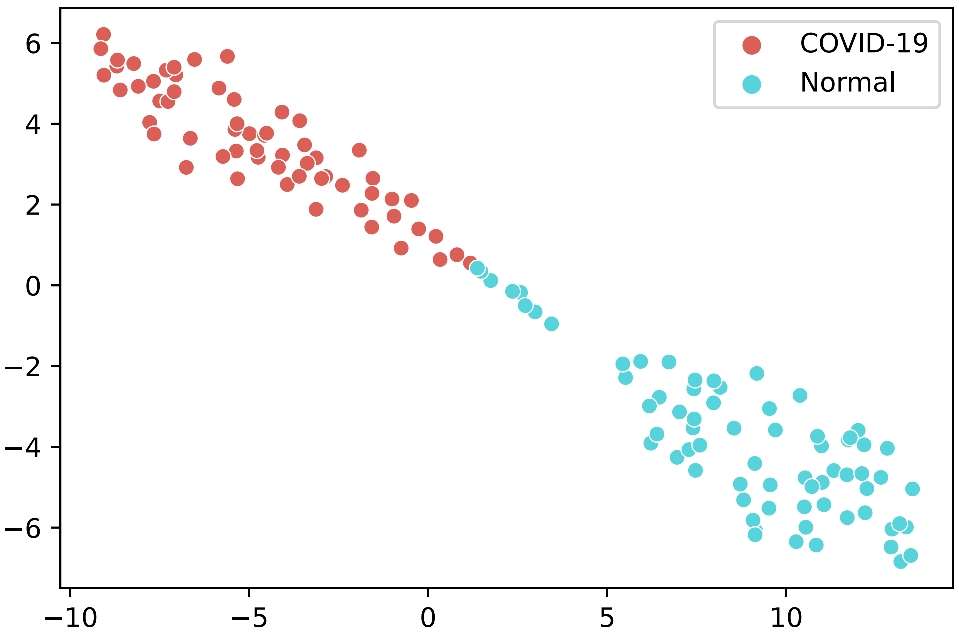
[10 points] TSNE Plot

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 [17]:

```
from sklearn.manifold import TSNE
import seaborn as sns
intermediate_layer_model = tf.keras.models.Model(inputs=model.input,
                                        outputs=model.get_layer('dense_feature').output)
tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                  batch_size=1,shuffle=False,seed=42,class_mode="binary")

intermediate_output = intermediate_layer_model.predict(tsne_data_generator)
tsne = TSNE(n_components=2)
tsne_data = tsne.fit_transform(intermediate_output)
labels = map(lambda x: "COVID-19" if x == 0 else "Normal", tsne_data_generator.labels)

scat=sns.scatterplot(tsne_data[:,0],tsne_data[:,1], hue=labels, palette=sns.color_palette("hls", 2))

plt.show()
#raise NotImplementedError("Extract features from the tsne_data_generator and fit a t-SNE model for the features,"
 #                          "and plot the resulting 2D features of the two classes.")
```

```
Found 130 images belonging to 2 classes.
/home/shawn/.local/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable
s as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```


● COVID-19
● Normal