task1

December 4, 2020

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

2 Task 1 [Total points: 30]

2.1 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
```

2.2 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.

2.3 [20 points] Binary Classification: COVID-19 vs. Normal

[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__
```

[1]: '2.3.1'

Load Image Data

[2]:
```
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

[3]:
```
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")
```

```
/Users/bellarocha/anaconda3/envs/tf2/lib/python3.7/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 '

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.

[4]:
```
vgg16 = tf.keras.applications.VGG16(
    include_top=False,
    weights="imagenet",
    input_shape= (224, 224, 3)
)

model = tf.keras.models.Sequential([
    vgg16,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation = 'relu', name = 'dense_feature'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation = 'sigmoid')
])

model.summary()
```

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

[5 points] Train Model

```
[5]: #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

OPTIMIZER = tf.keras.optimizers.SGD(learning_rate = LEARNING_RATE)

model.compile(optimizer = OPTIMIZER, loss = tf.keras.losses.
 ↪BinaryCrossentropy(), metrics = ['accuracy'])

history = model.fit(train_batches, epochs = NUM_EPOCHS, steps_per_epoch =␣
 ↪STEP_SIZE_TRAIN, validation_data = valid_batches, validation_steps =␣
 ↪STEP_SIZE_VALID)
```

```
11
3

/Users/bellarocha/anaconda3/envs/tf2/lib/python3.7/site-
packages/keras_preprocessing/image/image_data_generator.py:720: UserWarning:
This ImageDataGenerator specifies `featurewise_center`, but it hasn't been fit
on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/Users/bellarocha/anaconda3/envs/tf2/lib/python3.7/site-
packages/keras_preprocessing/image/image_data_generator.py:739: UserWarning:
This ImageDataGenerator specifies `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 [==============================] - 75s 7s/step - loss: 0.7663 - accuracy:
0.4574 - val_loss: 0.6746 - val_accuracy: 0.5500
Epoch 2/40
10/10 [==============================] - 69s 7s/step - loss: 0.6302 - accuracy:
0.6596 - val_loss: 0.5159 - val_accuracy: 0.9500
Epoch 3/40
10/10 [==============================] - 73s 7s/step - loss: 0.6430 - accuracy:
0.5700 - val_loss: 0.5000 - val_accuracy: 0.9000
Epoch 4/40
10/10 [==============================] - 80s 8s/step - loss: 0.6259 - accuracy:
0.6383 - val_loss: 0.5190 - val_accuracy: 0.6500
Epoch 5/40
10/10 [==============================] - 72s 7s/step - loss: 0.5463 - accuracy:
0.7553 - val_loss: 0.4607 - val_accuracy: 0.8500
Epoch 6/40
10/10 [==============================] - 75s 8s/step - loss: 0.4358 - accuracy:
0.7872 - val_loss: 0.4379 - val_accuracy: 0.9000
```

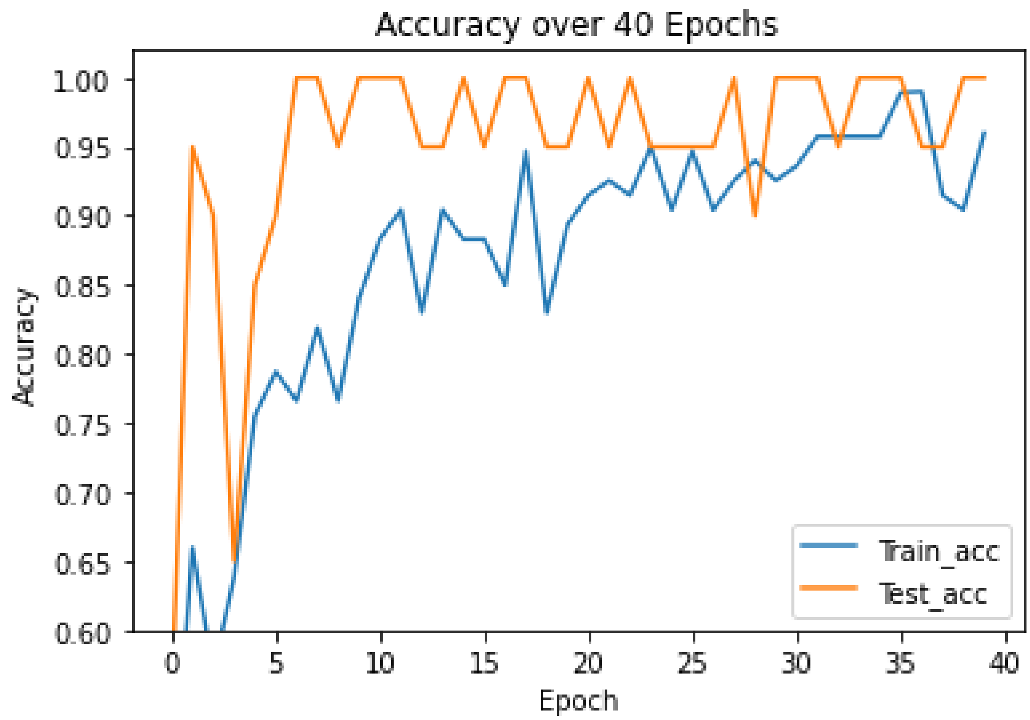
```
Epoch 7/40
10/10 [==============================] - 75s 7s/step - loss: 0.4806 - accuracy:
0.7660 - val_loss: 0.2739 - val_accuracy: 1.0000
Epoch 8/40
10/10 [==============================] - 76s 8s/step - loss: 0.4418 - accuracy:
0.8191 - val_loss: 0.2376 - val_accuracy: 1.0000
Epoch 9/40
10/10 [==============================] - 76s 8s/step - loss: 0.4260 - accuracy:
0.7660 - val_loss: 0.2071 - val_accuracy: 0.9500
Epoch 10/40
10/10 [==============================] - 77s 8s/step - loss: 0.3542 - accuracy:
0.8404 - val_loss: 0.2407 - val_accuracy: 1.0000
Epoch 11/40
10/10 [==============================] - 79s 8s/step - loss: 0.2748 - accuracy:
0.8830 - val_loss: 0.1238 - val_accuracy: 1.0000
Epoch 12/40
10/10 [==============================] - 73s 7s/step - loss: 0.2845 - accuracy:
0.9043 - val_loss: 0.1411 - val_accuracy: 1.0000
Epoch 13/40
10/10 [==============================] - 73s 7s/step - loss: 0.3900 - accuracy:
0.8298 - val_loss: 0.1437 - val_accuracy: 0.9500
Epoch 14/40
10/10 [==============================] - 73s 7s/step - loss: 0.2416 - accuracy:
0.9043 - val_loss: 0.1467 - val_accuracy: 0.9500
Epoch 15/40
10/10 [==============================] - 65s 7s/step - loss: 0.2489 - accuracy:
0.8830 - val_loss: 0.1153 - val_accuracy: 1.0000
Epoch 16/40
10/10 [==============================] - 65s 6s/step - loss: 0.2470 - accuracy:
0.8830 - val_loss: 0.0926 - val_accuracy: 0.9500
Epoch 17/40
10/10 [==============================] - 78s 8s/step - loss: 0.2997 - accuracy:
0.8500 - val_loss: 0.0941 - val_accuracy: 1.0000
Epoch 18/40
10/10 [==============================] - 65s 6s/step - loss: 0.1868 - accuracy:
0.9468 - val_loss: 0.0926 - val_accuracy: 1.0000
Epoch 19/40
10/10 [==============================] - 65s 7s/step - loss: 0.3663 - accuracy:
0.8298 - val_loss: 0.1077 - val_accuracy: 0.9500
Epoch 20/40
10/10 [==============================] - 65s 6s/step - loss: 0.2334 - accuracy:
0.8936 - val_loss: 0.0976 - val_accuracy: 0.9500
Epoch 21/40
10/10 [==============================] - 70s 7s/step - loss: 0.1820 - accuracy:
0.9149 - val_loss: 0.1140 - val_accuracy: 1.0000
Epoch 22/40
10/10 [==============================] - 69s 7s/step - loss: 0.1869 - accuracy:
0.9255 - val_loss: 0.1229 - val_accuracy: 0.9500
```

```
Epoch 23/40
10/10 [==============================] - 65s 7s/step - loss: 0.1670 - accuracy:
0.9149 - val_loss: 0.0460 - val_accuracy: 1.0000
Epoch 24/40
10/10 [==============================] - 69s 7s/step - loss: 0.1556 - accuracy:
0.9500 - val_loss: 0.0847 - val_accuracy: 0.9500
Epoch 25/40
10/10 [==============================] - 65s 6s/step - loss: 0.2279 - accuracy:
0.9043 - val_loss: 0.1085 - val_accuracy: 0.9500
Epoch 26/40
10/10 [==============================] - 65s 7s/step - loss: 0.1304 - accuracy:
0.9468 - val_loss: 0.0629 - val_accuracy: 0.9500
Epoch 27/40
10/10 [==============================] - 65s 6s/step - loss: 0.1960 - accuracy:
0.9043 - val_loss: 0.1150 - val_accuracy: 0.9500
Epoch 28/40
10/10 [==============================] - 65s 6s/step - loss: 0.2079 - accuracy:
0.9255 - val_loss: 0.0500 - val_accuracy: 1.0000
Epoch 29/40
10/10 [==============================] - 69s 7s/step - loss: 0.1393 - accuracy:
0.9400 - val_loss: 0.1353 - val_accuracy: 0.9000
Epoch 30/40
10/10 [==============================] - 65s 7s/step - loss: 0.1785 - accuracy:
0.9255 - val_loss: 0.0658 - val_accuracy: 1.0000
Epoch 31/40
10/10 [==============================] - 65s 7s/step - loss: 0.1428 - accuracy:
0.9362 - val_loss: 0.0562 - val_accuracy: 1.0000
Epoch 32/40
10/10 [==============================] - 69s 7s/step - loss: 0.1050 - accuracy:
0.9574 - val_loss: 0.0490 - val_accuracy: 1.0000
Epoch 33/40
10/10 [==============================] - 68s 7s/step - loss: 0.1268 - accuracy:
0.9574 - val_loss: 0.0828 - val_accuracy: 0.9500
Epoch 34/40
10/10 [==============================] - 65s 6s/step - loss: 0.1082 - accuracy:
0.9574 - val_loss: 0.0233 - val_accuracy: 1.0000
Epoch 35/40
10/10 [==============================] - 65s 6s/step - loss: 0.1178 - accuracy:
0.9574 - val_loss: 0.0404 - val_accuracy: 1.0000
Epoch 36/40
10/10 [==============================] - 64s 6s/step - loss: 0.0662 - accuracy:
0.9894 - val_loss: 0.0327 - val_accuracy: 1.0000
Epoch 37/40
10/10 [==============================] - 69s 7s/step - loss: 0.0703 - accuracy:
0.9900 - val_loss: 0.0570 - val_accuracy: 0.9500
Epoch 38/40
10/10 [==============================] - 65s 6s/step - loss: 0.1673 - accuracy:
0.9149 - val_loss: 0.1675 - val_accuracy: 0.9500
```

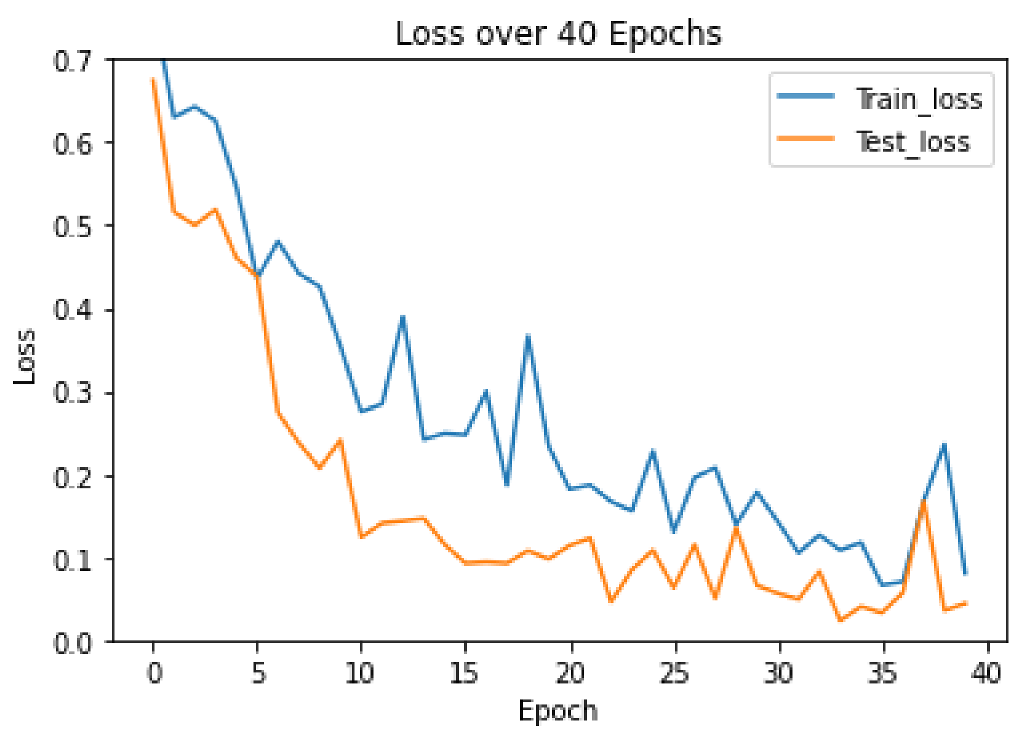
```
Epoch 39/40
10/10 [==============================] - 65s 6s/step - loss: 0.2361 - accuracy:
0.9043 - val_loss: 0.0356 - val_accuracy: 1.0000
Epoch 40/40
10/10 [==============================] - 69s 7s/step - loss: 0.0806 - accuracy:
0.9600 - val_loss: 0.0440 - val_accuracy: 1.0000
```

[5 points] Plot Accuracy and Loss During Training

[9]:
```python
import matplotlib.pyplot as plt

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, 1.02])
plt.legend(loc='lower right')
plt.title('Accuracy over 40 Epochs')
plt.show()

plt.plot(history.history['loss'], label='Train_loss')
plt.plot(history.history['val_loss'], label = 'Test_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.ylim([0, 0.7])
plt.legend(loc='upper right')
plt.title('Loss over 40 Epochs')
plt.show()
```



Accuracy over 40 Epochs
Accuracy
Epoch
Train_acc
Test_acc

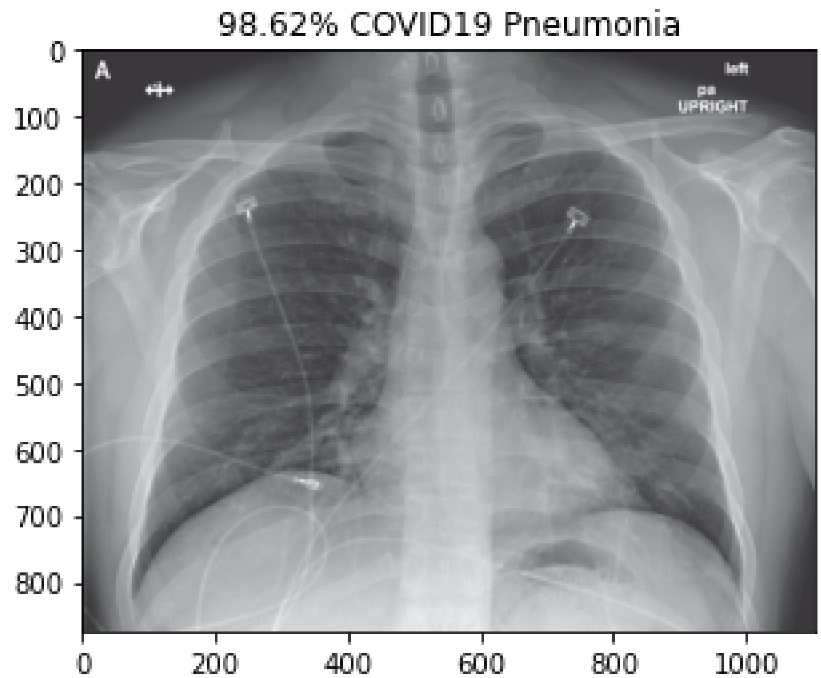


Loss over 40 Epochs
Loss
Epoch
Train_loss
Test_loss

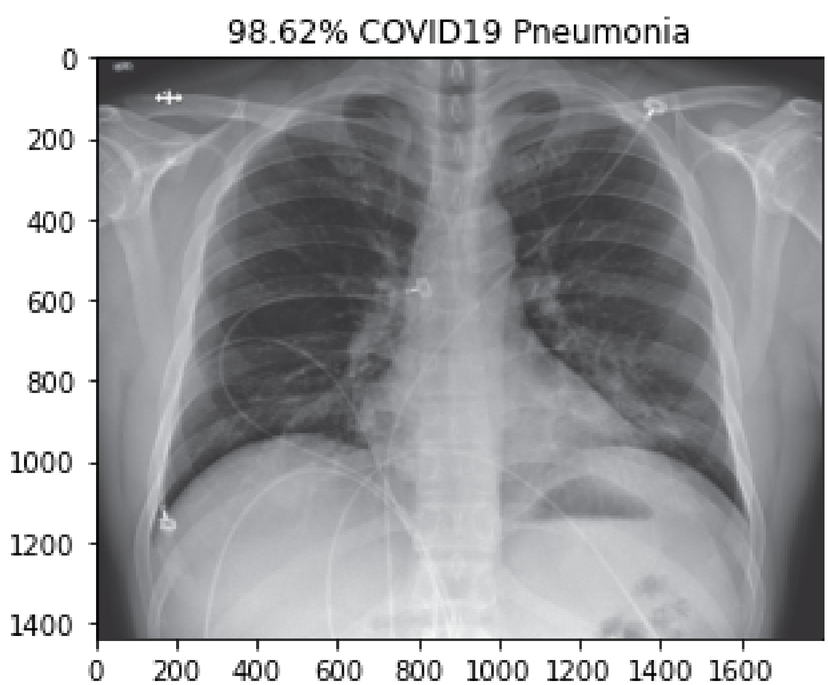
Plot Test Results

[10]:
```
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()
```

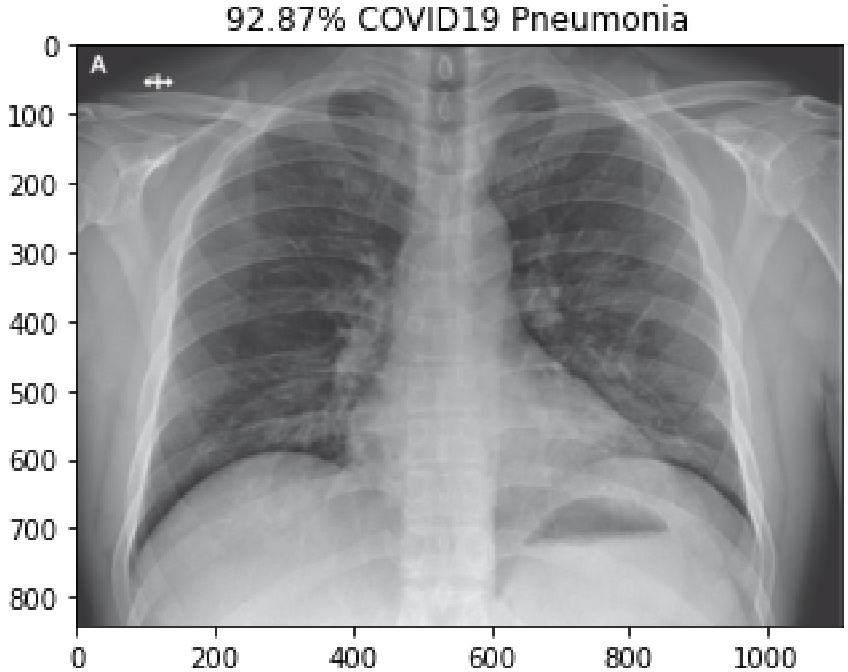
```
Found 18 images belonging to 2 classes.
18/18 [==============================] - 4s 207ms/step
covid/nejmoa2001191_f3-PA.jpeg
```



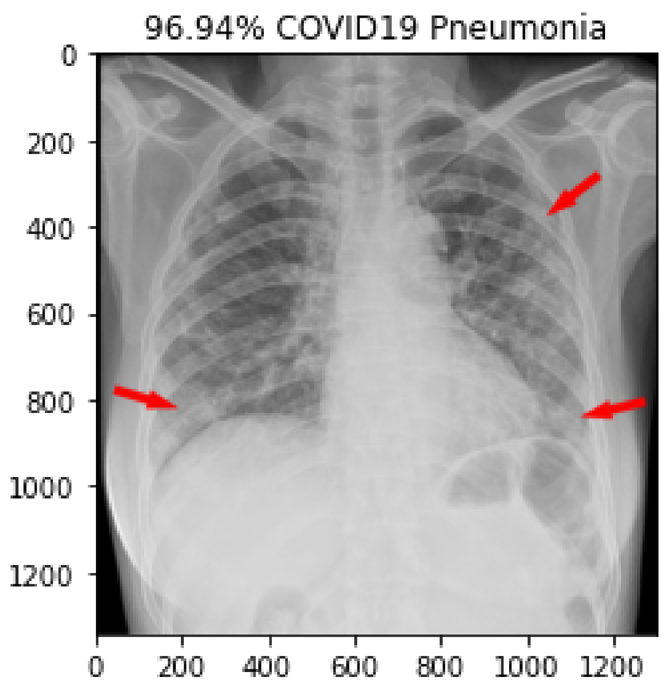
covid/nejmoa2001191_f4.jpeg



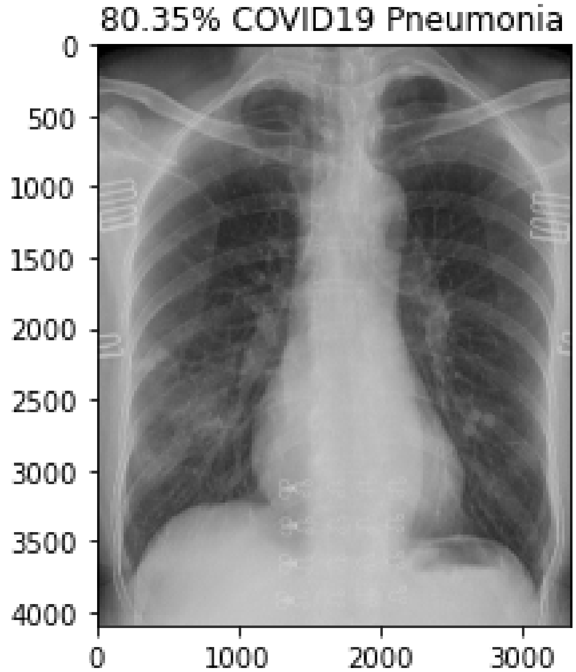
covid/nejmoa2001191_f5-PA.jpeg



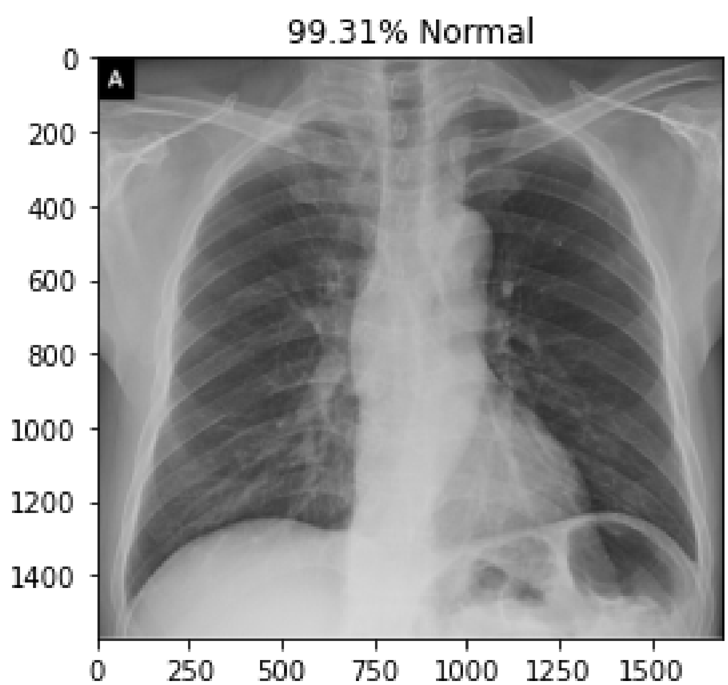
covid/radiol.2020200490.fig3.jpeg



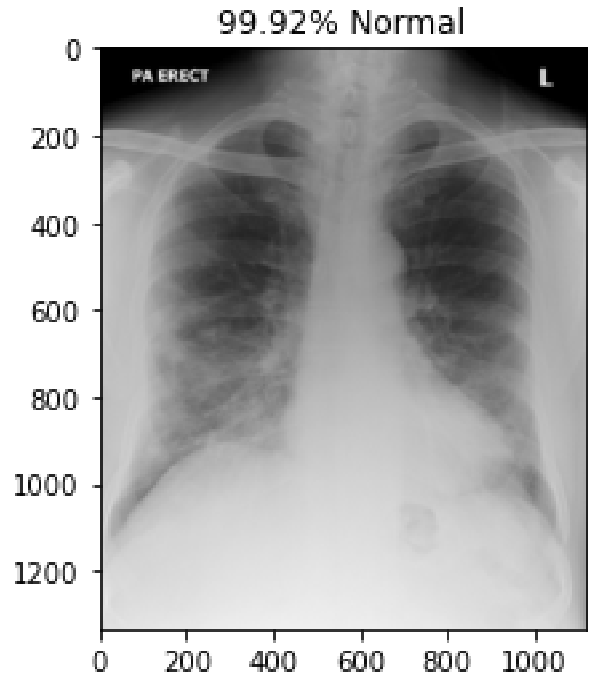
covid/ryct.2020200028.fig1a.jpeg



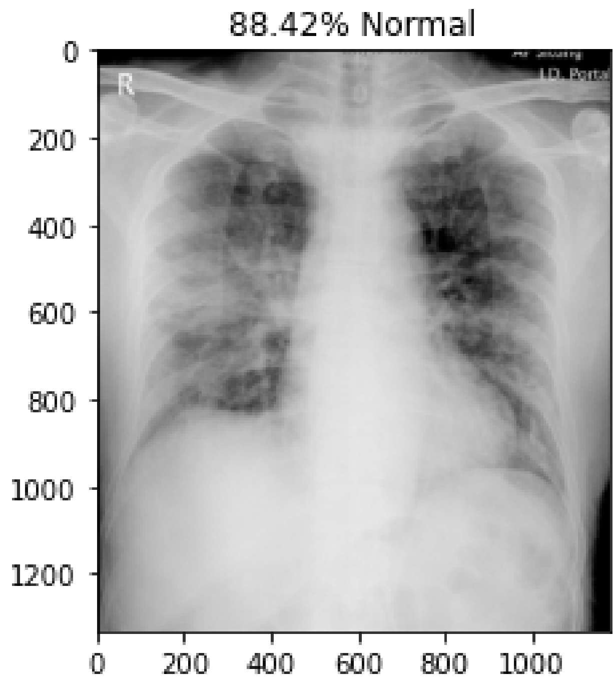
covid/ryct.2020200034.fig2.jpeg



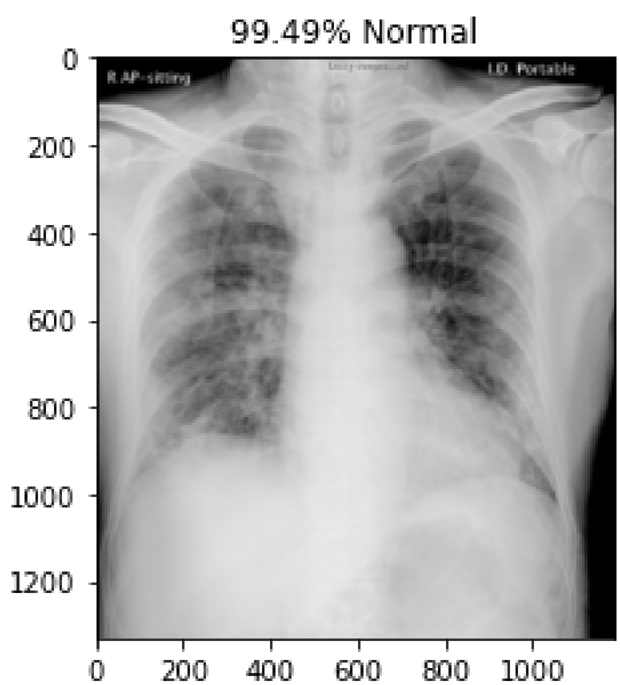
covid/ryct.2020200034.fig5-day0.jpeg



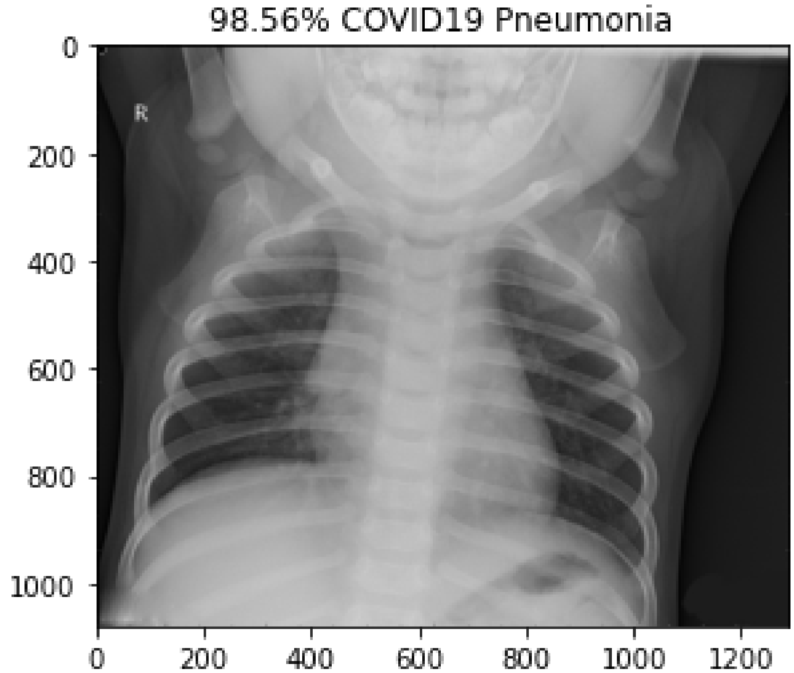
covid/ryct.2020200034.fig5-day4.jpeg



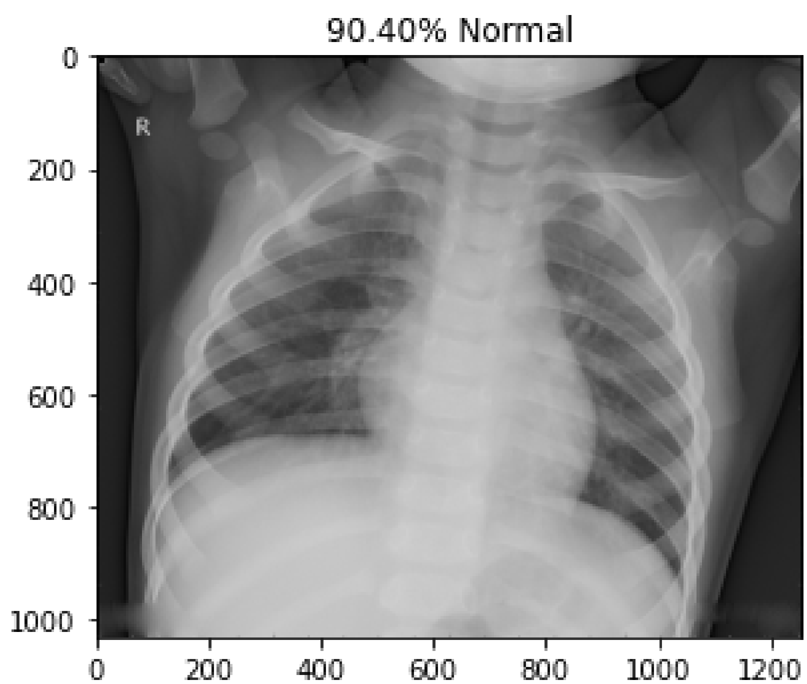
covid/ryct.2020200034.fig5-day7.jpeg



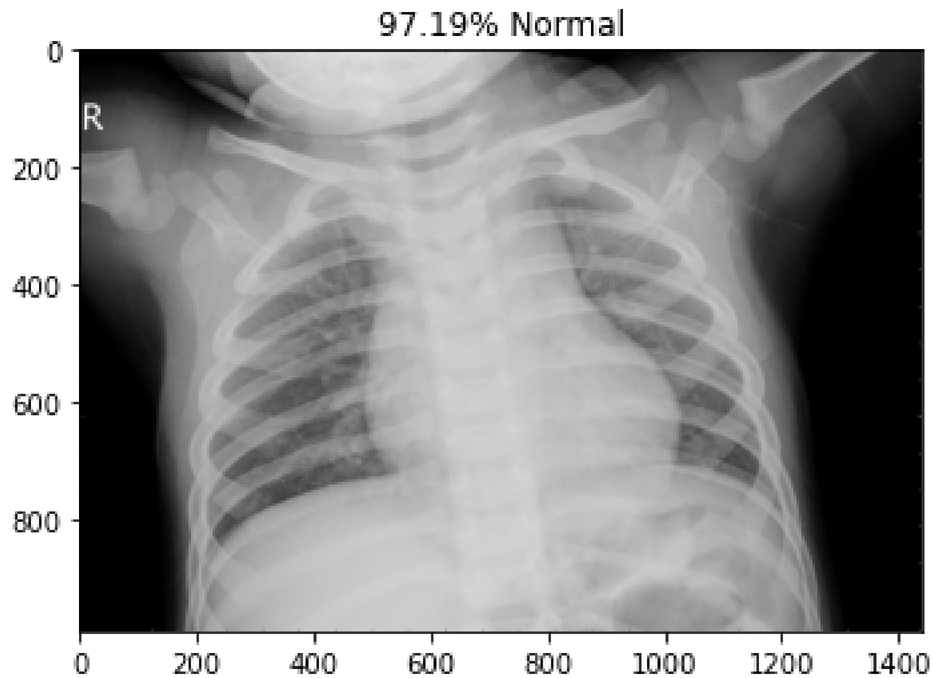
normal/NORMAL2-IM-1385-0001.jpeg



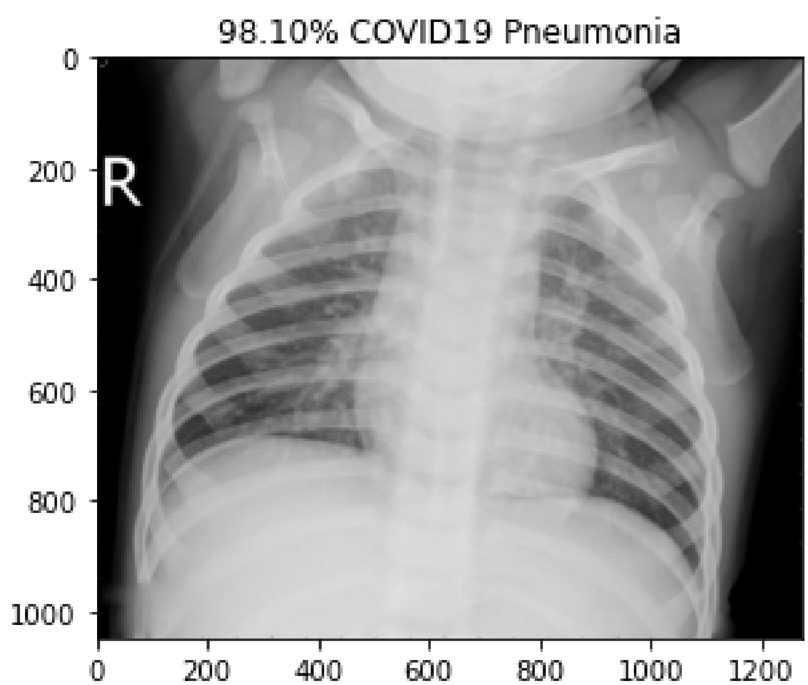
normal/NORMAL2-IM-1396-0001.jpeg



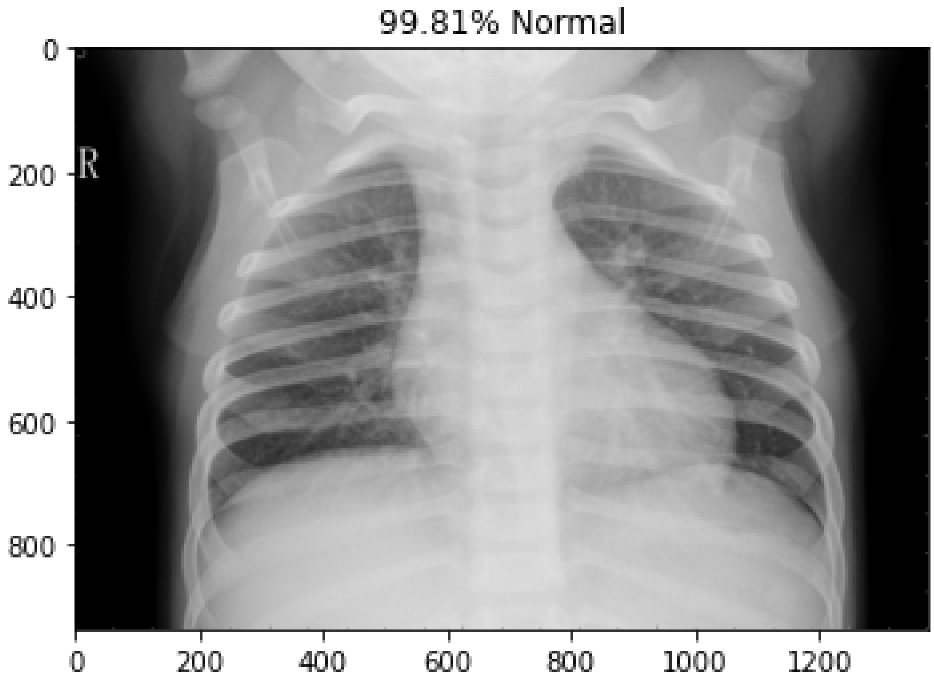
```
normal/NORMAL2-IM-1400-0001.jpeg
```



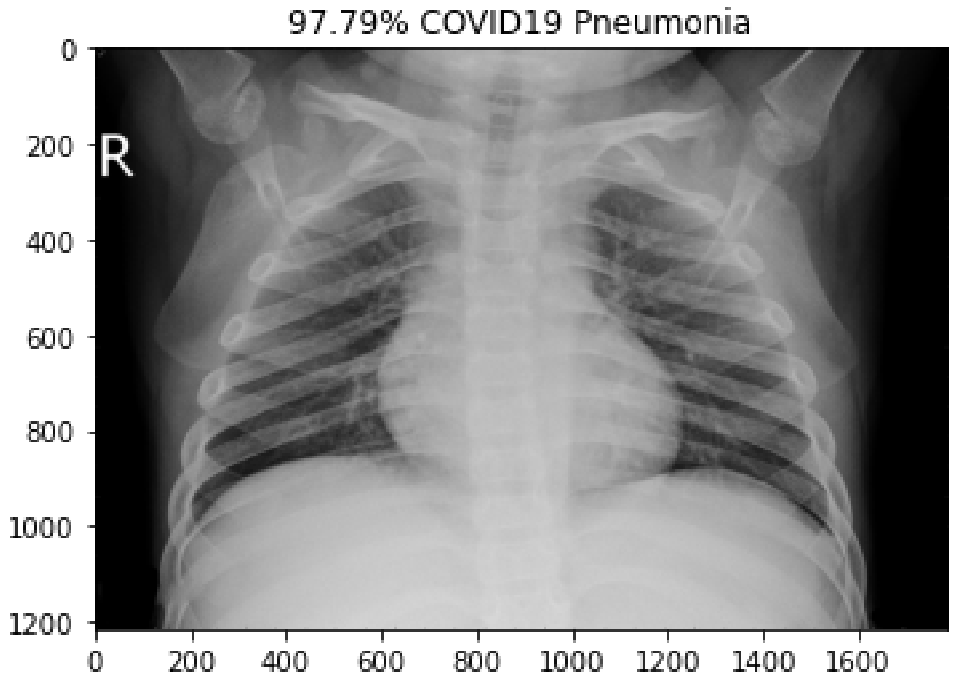

```
normal/NORMAL2-IM-1401-0001.jpeg
```



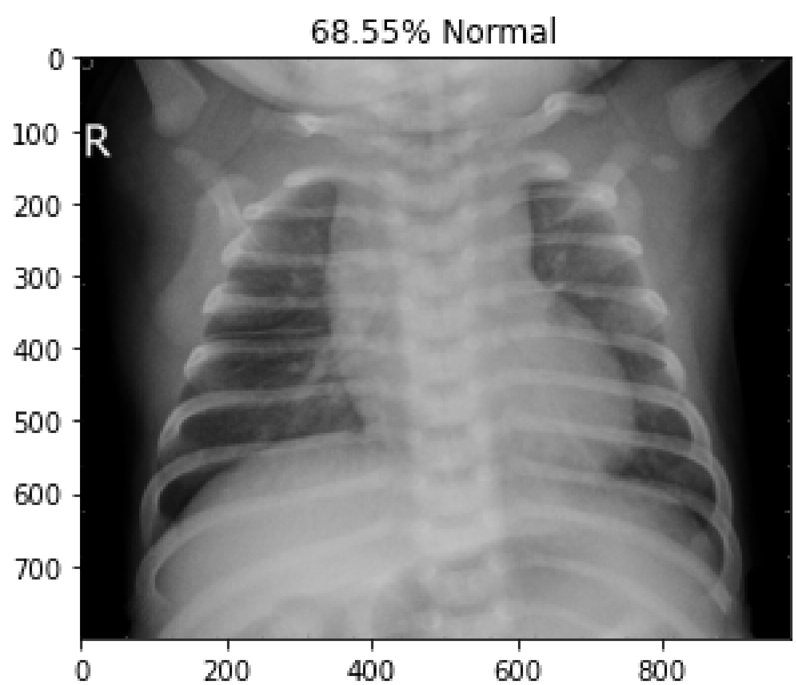
normal/NORMAL2-IM-1406-0001.jpeg



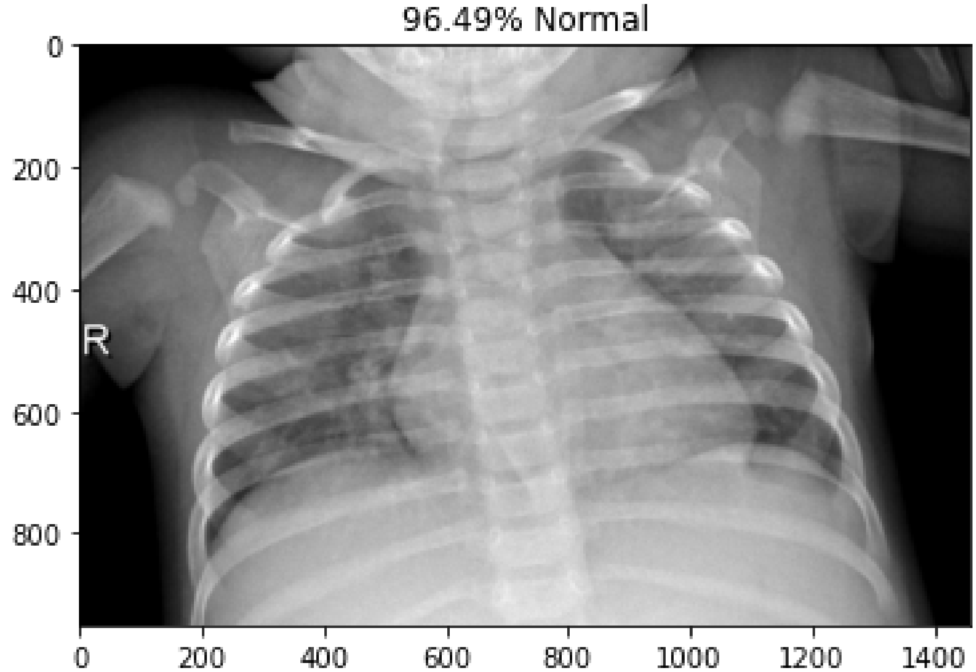
normal/NORMAL2-IM-1412-0001.jpeg



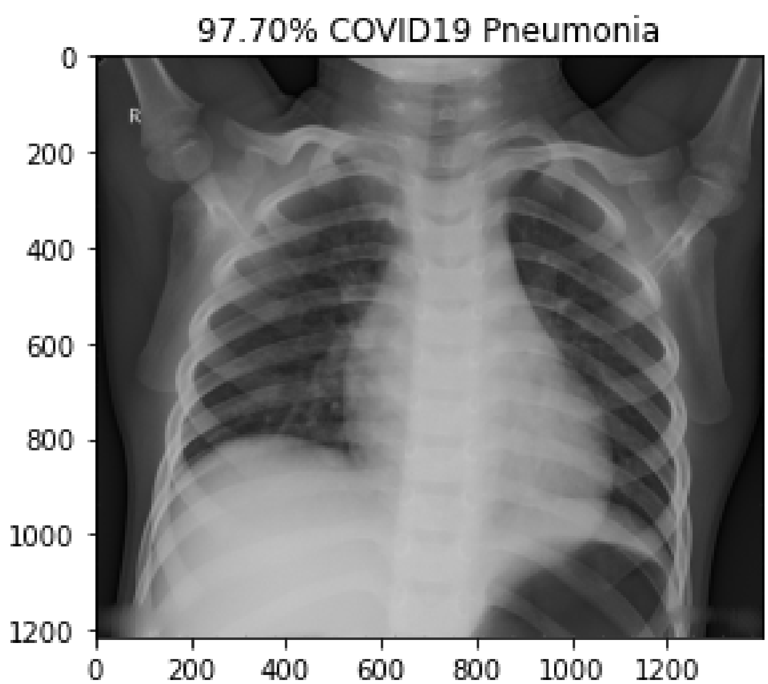
normal/NORMAL2-IM-1419-0001.jpeg



normal/NORMAL2-IM-1422-0001.jpeg



normal/NORMAL2-IM-1423-0001.jpeg

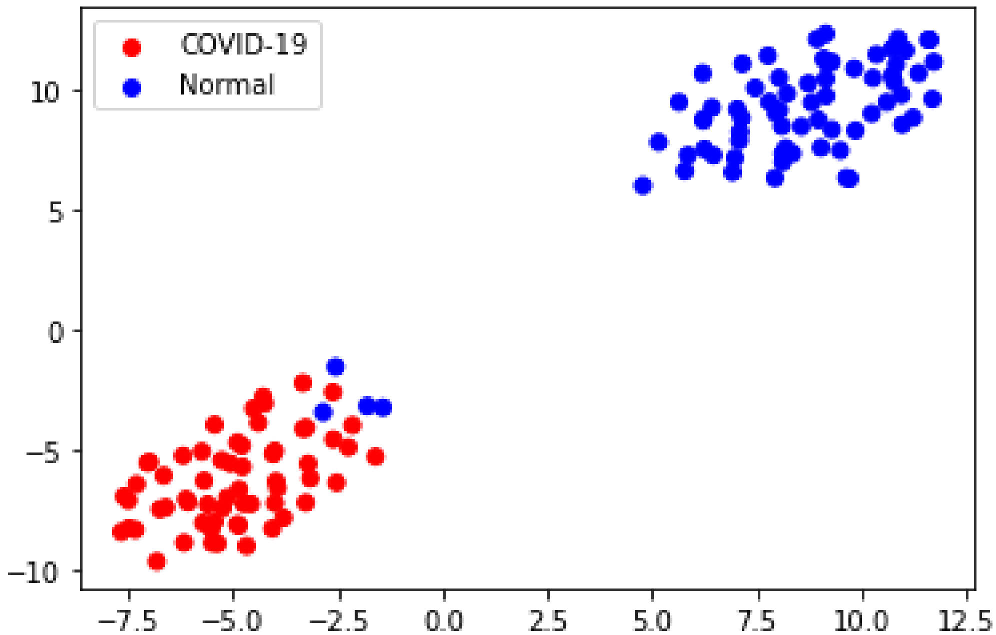


2.4 [10 points] TSNE Plot

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.

[18]:
```
from sklearn.manifold import TSNE

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_layer = intermediate_layer_model.predict(tsne_data_generator)

intermediate_layer_TSNE = TSNE().fit_transform(intermediate_layer)

classes = tsne_data_generator.classes

colors = []

for i in range(len(classes)):
    if classes[i] == 0:
        colors.append('red')
    else:
        colors.append('blue')

plt.scatter(intermediate_layer_TSNE[:, 0], intermediate_layer_TSNE[:, 1], color␣
 ↪= colors)
plt.scatter(intermediate_layer_TSNE[:, 0][0], intermediate_layer_TSNE[:, 1][0],␣
 ↪color = colors[0], label = 'COVID-19')
plt.scatter(intermediate_layer_TSNE[:, 0][70], intermediate_layer_TSNE[:,␣
 ↪1][70], color = colors[70], label = 'Normal')
plt.legend()
plt.show()
```

```
Found 130 images belonging to 2 classes.
```


COVID-19
Normal
10
5
0
−5
−10
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5