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.

In [4]:

```
from google.colab import drive
drive.mount('/content/drive')
```

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

In [ ]:

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
 raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

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

In [ ]:

```
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[ ]:

```
'2.8.0'
```

Load Image Data

In [ ]:

```
DATA_LIST = os.listdir('/content/drive/MyDrive/challenge/two/train')
DATASET_PATH  = '/content/drive/MyDrive/challenge/two/train'
TEST_DIR =  '/content/drive/MyDrive/challenge/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 wit
h reducing it gradually
```

Generate Training and Validation Batches

In [ ]:

```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_
center = True,
                                   featurewise_std_normalization = True,width_sh
ift_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.
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/ima
ge_data_generator.py:342: UserWarning: This ImageDataGenerator speci
fies `zca_whitening` which overrides setting of`featurewise_std_norm
alization`.
  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 [ ]:

```
from tensorflow.keras import Sequential,layers
from keras import models
from keras.models import *
from keras.layers import *
from keras.preprocessing.image import *
from keras.utils import *
from keras.optimizers import *
from keras.applications import *
from keras.applications import  imagenet_utils
from keras.callbacks import EarlyStopping,ReduceLROnPlateau,ModelCheckpoint,Lear
ningRateScheduler
from keras.applications import vgg16


# 模型 VGG161.0

base_model = vgg16.VGG16(weights="imagenet",include_top=False,input_shape=(224,2
24,3))
base_model.trainable = False
x = base_model.output
# x = GlobalAveragePooling2D()(x)
# x = Dense(128,activation="relu")(x)
# x = Dropout(0.5)(x)
x = layers.Flatten()(x)
# x = GlobalAveragePooling2D()(x)

x = Dense(256, name='dense_feature')(x)

out = Dense(1,activation="sigmoid")(x)
model = Model(base_model.input,out)

# model = models.Sequential()
# model.add(base_model)
# model.add(layers.Flatten())
# model.add(layers.Dense(256, activation='relu', name='dense_feature'))
# model.add(layers.Dropout(0.1))
# model.add(layers.Dense(1, activation='sigmoid'))

# model.build(input_shape=(224, 224, 3))
model.summary()
```

```
Model: "model_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 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
_________________________________________________________________
```

In [ ]:

```
import keras.backend as K
from keras import optimizers

# focal loss
def focal_loss(alpha=0.25,gamma=2.0):
    def focal_crossentropy(y_true, y_pred):
        bce = K.binary_crossentropy(y_true, y_pred)

        y_pred = K.clip(y_pred, K.epsilon(), 1.- K.epsilon())
        p_t = (y_true*y_pred) + ((1-y_true)*(1-y_pred))

        alpha_factor = 1
        modulating_factor = 1

        alpha_factor = y_true*alpha + ((1-alpha)*(1-y_true))
        modulating_factor = K.pow((1-p_t), gamma)

        # compute the final loss and return
        return K.mean(alpha_factor*modulating_factor*bce, axis=-1)
    return focal_crossentropy

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0005),loss=foca
l_loss(),metrics=["accuracy"])
```

[5 points] Train Model

In [ ]:

```
#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

history = model.fit_generator(train_batches,
                              steps_per_epoch=STEP_SIZE_TRAIN,
                                 epochs = 40,
                                 validation_data=valid_batches,
                                 validation_steps=STEP_SIZE_VALID,
                                 shuffle=True)
# raise NotImplementedError("Use the model.fit function to train your network")
```

```
11
3

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: Use
rWarning: `Model.fit_generator` is deprecated and will be removed in
a future version. Please use `Model.fit`, which supports generators.
  del sys.path[0]
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/ima
ge_data_generator.py:720: UserWarning: This ImageDataGenerator speci
fies `featurewise_center`, but it hasn't been fit on any training da
ta. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/ima
ge_data_generator.py:739: UserWarning: This ImageDataGenerator speci
fies `zca_whitening`, but it hasn't been fit on any training data. F
it it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
```

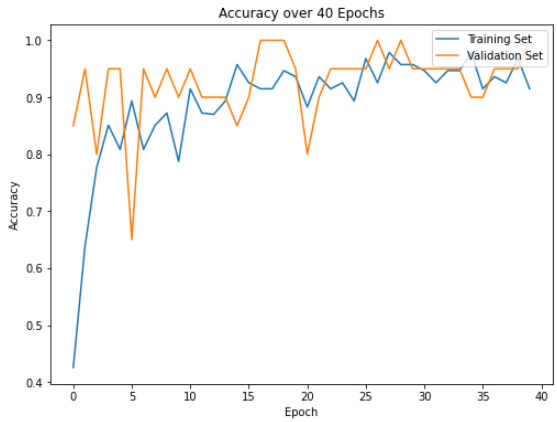
```
Epoch 1/40
10/10 [==============================] - 7s 632ms/step - loss: 3.377
6 - accuracy: 0.4255 - val_loss: 0.1022 - val_accuracy: 0.8500
Epoch 2/40
10/10 [==============================] - 5s 532ms/step - loss: 0.958
4 - accuracy: 0.6383 - val_loss: 0.2949 - val_accuracy: 0.9500
Epoch 3/40
10/10 [==============================] - 5s 550ms/step - loss: 0.501
9 - accuracy: 0.7766 - val_loss: 0.3791 - val_accuracy: 0.8000
Epoch 4/40
10/10 [==============================] - 6s 551ms/step - loss: 0.357
8 - accuracy: 0.8511 - val_loss: 0.0221 - val_accuracy: 0.9500
Epoch 5/40
10/10 [==============================] - 6s 575ms/step - loss: 0.286
9 - accuracy: 0.8085 - val_loss: 0.0081 - val_accuracy: 0.9500
Epoch 6/40
10/10 [==============================] - 6s 562ms/step - loss: 0.226
7 - accuracy: 0.8936 - val_loss: 0.4203 - val_accuracy: 0.6500
Epoch 7/40
10/10 [==============================] - 5s 557ms/step - loss: 0.327
3 - accuracy: 0.8085 - val_loss: 0.0273 - val_accuracy: 0.9500
Epoch 8/40
10/10 [==============================] - 6s 537ms/step - loss: 0.334
0 - accuracy: 0.8511 - val_loss: 0.0398 - val_accuracy: 0.9000
Epoch 9/40
10/10 [==============================] - 6s 560ms/step - loss: 0.233
7 - accuracy: 0.8723 - val_loss: 0.0519 - val_accuracy: 0.9500
Epoch 10/40
10/10 [==============================] - 6s 564ms/step - loss: 0.347
9 - accuracy: 0.7872 - val_loss: 0.2649 - val_accuracy: 0.9000
Epoch 11/40
10/10 [==============================] - 5s 561ms/step - loss: 0.226
4 - accuracy: 0.9149 - val_loss: 0.0138 - val_accuracy: 0.9500
Epoch 12/40
10/10 [==============================] - 6s 530ms/step - loss: 0.154
8 - accuracy: 0.8723 - val_loss: 0.5866 - val_accuracy: 0.9000
Epoch 13/40
10/10 [==============================] - 6s 580ms/step - loss: 0.242
4 - accuracy: 0.8700 - val_loss: 0.1068 - val_accuracy: 0.9000
Epoch 14/40
10/10 [==============================] - 6s 566ms/step - loss: 0.158
8 - accuracy: 0.8936 - val_loss: 0.4080 - val_accuracy: 0.9000
Epoch 15/40
10/10 [==============================] - 6s 619ms/step - loss: 0.107
9 - accuracy: 0.9574 - val_loss: 0.0516 - val_accuracy: 0.8500
Epoch 16/40
10/10 [==============================] - 6s 566ms/step - loss: 0.101
7 - accuracy: 0.9255 - val_loss: 0.9081 - val_accuracy: 0.9000
Epoch 17/40
10/10 [==============================] - 5s 552ms/step - loss: 0.101
6 - accuracy: 0.9149 - val_loss: 1.6950e-04 - val_accuracy: 1.0000
Epoch 18/40
10/10 [==============================] - 5s 538ms/step - loss: 0.241
4 - accuracy: 0.9149 - val_loss: 3.4810e-05 - val_accuracy: 1.0000
Epoch 19/40
10/10 [==============================] - 5s 570ms/step - loss: 0.098
9 - accuracy: 0.9468 - val_loss: 1.6701e-04 - val_accuracy: 1.0000
Epoch 20/40
10/10 [==============================] - 5s 527ms/step - loss: 0.109
4 - accuracy: 0.9362 - val_loss: 0.0066 - val_accuracy: 0.9500
Epoch 21/40
```

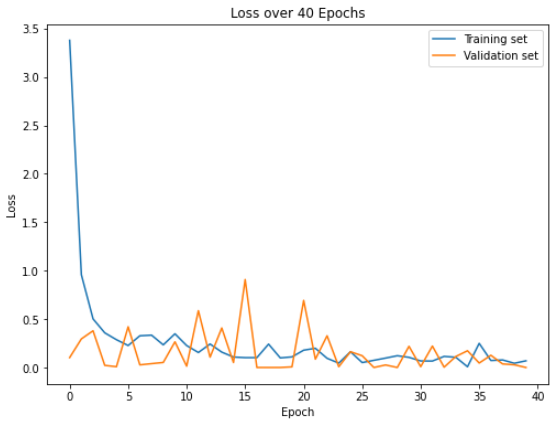
```
10/10 [==============================] - 5s 533ms/step - loss: 0.178
6 - accuracy: 0.8830 - val_loss: 0.6926 - val_accuracy: 0.8000
Epoch 22/40
10/10 [==============================] - 6s 557ms/step - loss: 0.198
3 - accuracy: 0.9362 - val_loss: 0.0858 - val_accuracy: 0.9000
Epoch 23/40
10/10 [==============================] - 6s 557ms/step - loss: 0.094
8 - accuracy: 0.9149 - val_loss: 0.3274 - val_accuracy: 0.9500
Epoch 24/40
10/10 [==============================] - 5s 552ms/step - loss: 0.045
0 - accuracy: 0.9255 - val_loss: 0.0078 - val_accuracy: 0.9500
Epoch 25/40
10/10 [==============================] - 6s 590ms/step - loss: 0.162
8 - accuracy: 0.8936 - val_loss: 0.1629 - val_accuracy: 0.9500
Epoch 26/40
10/10 [==============================] - 5s 565ms/step - loss: 0.050
7 - accuracy: 0.9681 - val_loss: 0.1214 - val_accuracy: 0.9500
Epoch 27/40
10/10 [==============================] - 6s 562ms/step - loss: 0.073
9 - accuracy: 0.9255 - val_loss: 7.2357e-07 - val_accuracy: 1.0000
Epoch 28/40
10/10 [==============================] - 5s 547ms/step - loss: 0.097
1 - accuracy: 0.9787 - val_loss: 0.0262 - val_accuracy: 0.9500
Epoch 29/40
10/10 [==============================] - 6s 586ms/step - loss: 0.121
6 - accuracy: 0.9574 - val_loss: 3.1650e-08 - val_accuracy: 1.0000
Epoch 30/40
10/10 [==============================] - 5s 543ms/step - loss: 0.104
8 - accuracy: 0.9574 - val_loss: 0.2188 - val_accuracy: 0.9500
Epoch 31/40
10/10 [==============================] - 6s 553ms/step - loss: 0.067
5 - accuracy: 0.9468 - val_loss: 0.0084 - val_accuracy: 0.9500
Epoch 32/40
10/10 [==============================] - 6s 524ms/step - loss: 0.066
4 - accuracy: 0.9255 - val_loss: 0.2213 - val_accuracy: 0.9500
Epoch 33/40
10/10 [==============================] - 5s 540ms/step - loss: 0.114
5 - accuracy: 0.9468 - val_loss: 0.0030 - val_accuracy: 0.9500
Epoch 34/40
10/10 [==============================] - 6s 547ms/step - loss: 0.105
6 - accuracy: 0.9468 - val_loss: 0.1134 - val_accuracy: 0.9500
Epoch 35/40
10/10 [==============================] - 5s 546ms/step - loss: 0.007
8 - accuracy: 0.9787 - val_loss: 0.1729 - val_accuracy: 0.9000
Epoch 36/40
10/10 [==============================] - 5s 583ms/step - loss: 0.249
4 - accuracy: 0.9149 - val_loss: 0.0460 - val_accuracy: 0.9000
Epoch 37/40
10/10 [==============================] - 5s 546ms/step - loss: 0.071
8 - accuracy: 0.9362 - val_loss: 0.1263 - val_accuracy: 0.9500
Epoch 38/40
10/10 [==============================] - 5s 550ms/step - loss: 0.077
9 - accuracy: 0.9255 - val_loss: 0.0367 - val_accuracy: 0.9500
Epoch 39/40
10/10 [==============================] - 5s 548ms/step - loss: 0.043
6 - accuracy: 0.9681 - val_loss: 0.0281 - val_accuracy: 0.9500
Epoch 40/40
10/10 [==============================] - 5s 531ms/step - loss: 0.069
0 - accuracy: 0.9149 - val_loss: 1.3706e-06 - val_accuracy: 1.0000
```

[5 points] Plot Accuracy and Loss During Training

In [ ]:

```
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
%matplotlib inline

# #Accuracy
plt.figure(figsize=(18,6))
plt.subplot(1,2,1)
plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
plt.title("Accuracy over 40 Epochs")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(["Training Set","Validation Set"],loc="upper right")
# Loss
plt.subplot(1,2,2)
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Loss over 40 Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(["Training set","Validation set"],loc="upper right")

plt.show()

# raise NotImplementedError("Plot the accuracy and the loss during training")
```



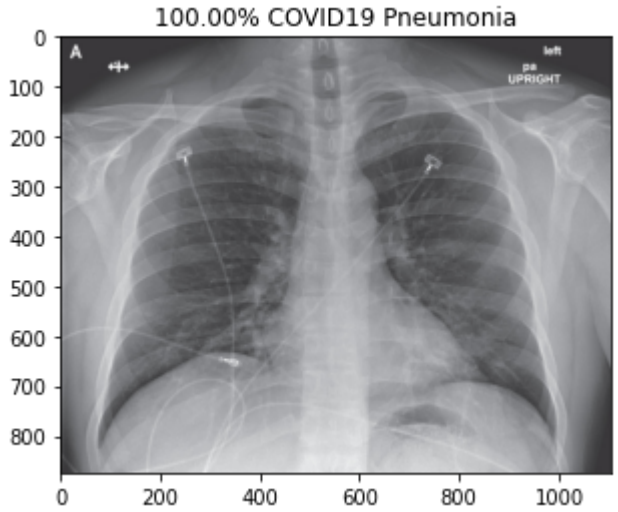


Plot Test Results

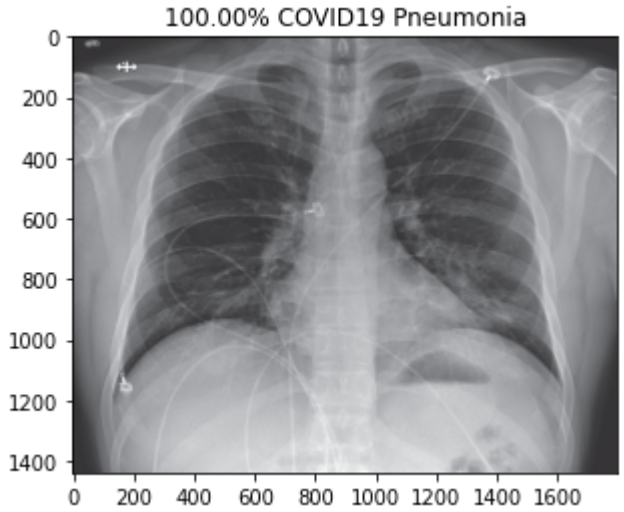
In [ ]:

```
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()
```

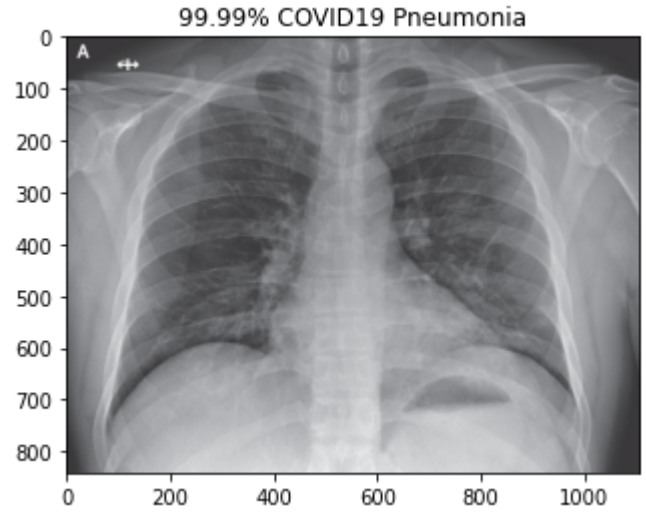
```
Found 18 images belonging to 2 classes.

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: User
Warning: `Model.predict_generator` is deprecated and will be removed
in a future version. Please use `Model.predict`, which supports gene
rators.
  import sys

18/18 [==============================] - 1s 45ms/step
covid/nejmoa2001191_f3-PA.jpeg
```



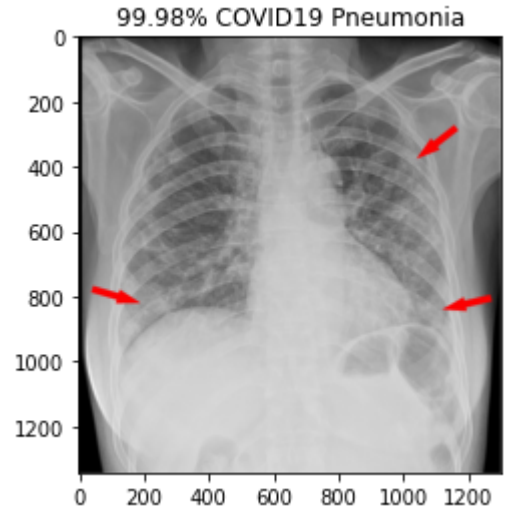

```
covid/nejmoa2001191_f4.jpeg
```



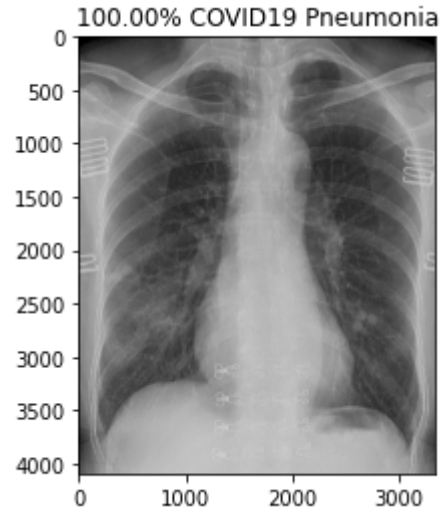

```
covid/nejmoa2001191_f5-PA.jpeg
```



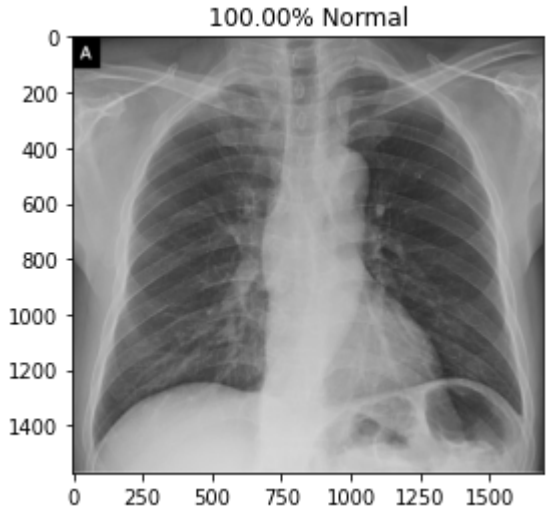
covid/radiol.2020200490.fig3.jpeg



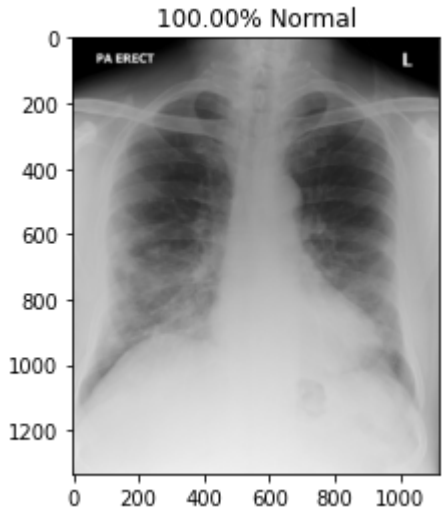
covid/ryct.2020200028.fig1a.jpeg



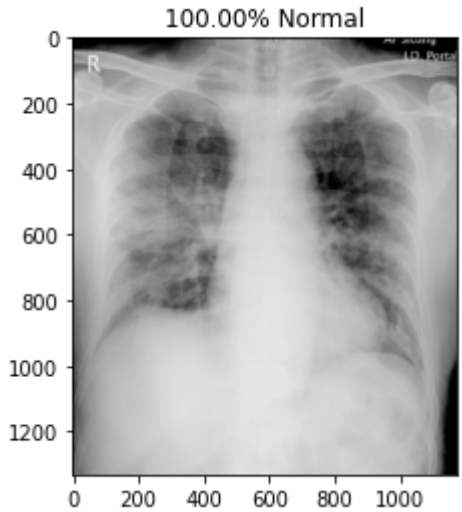
covid/ryct.2020200034.fig2.jpeg



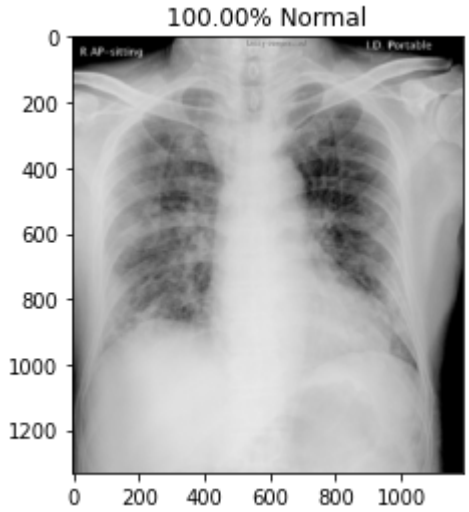
covid/ryct.2020200034.fig5-day0.jpeg



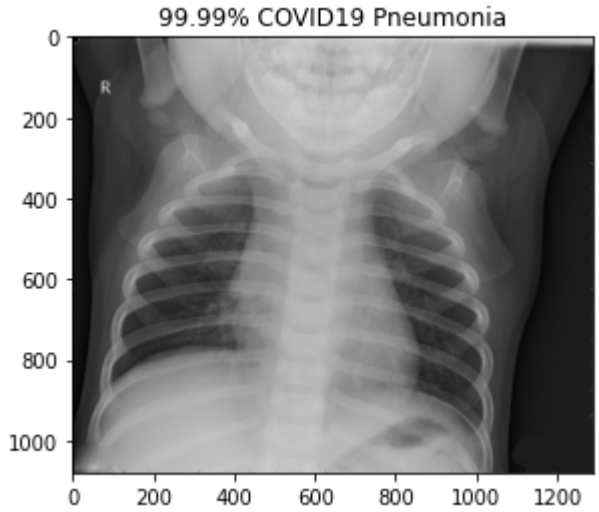
covid/ryct.2020200034.fig5-day4.jpeg



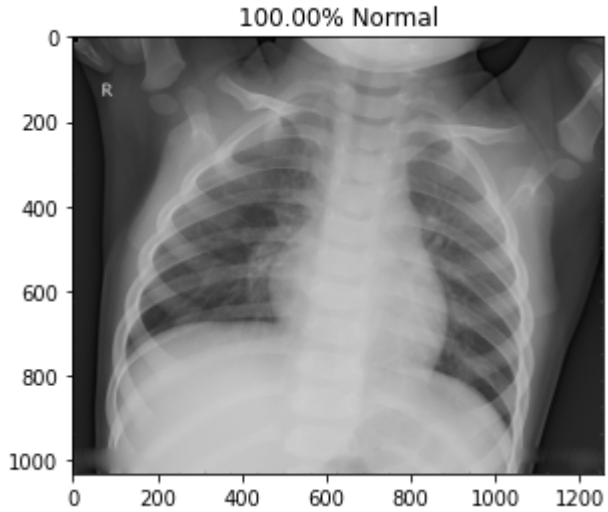
covid/ryct.2020200034.fig5-day7.jpeg



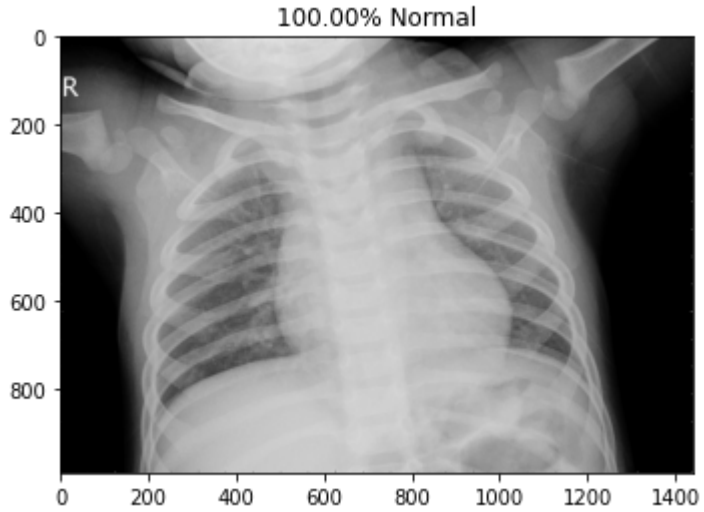
normal/NORMAL2-IM-1385-0001.jpeg



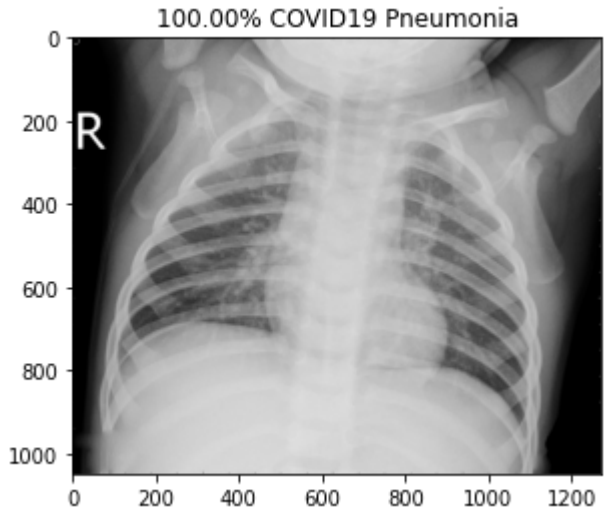
normal/NORMAL2-IM-1396-0001.jpeg



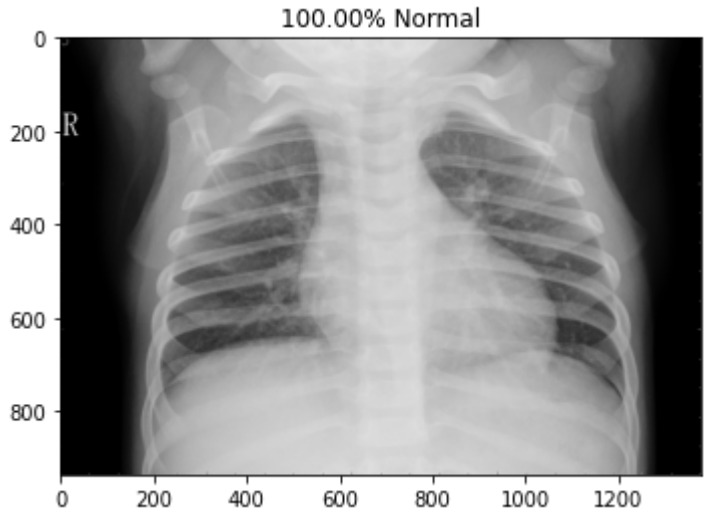
normal/NORMAL2-IM-1400-0001.jpeg



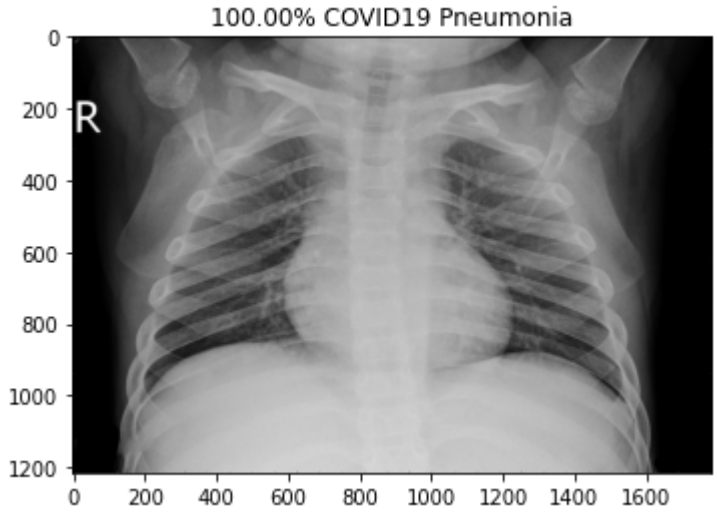
normal/NORMAL2-IM-1401-0001.jpeg



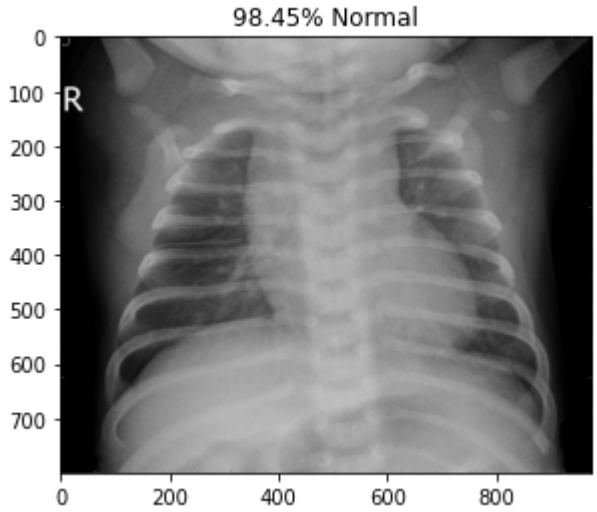
normal/NORMAL2-IM-1406-0001.jpeg



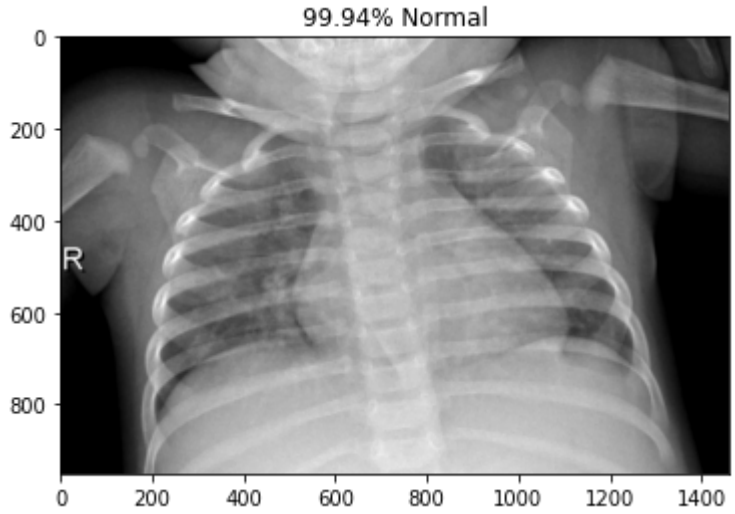
normal/NORMAL2-IM-1412-0001.jpeg



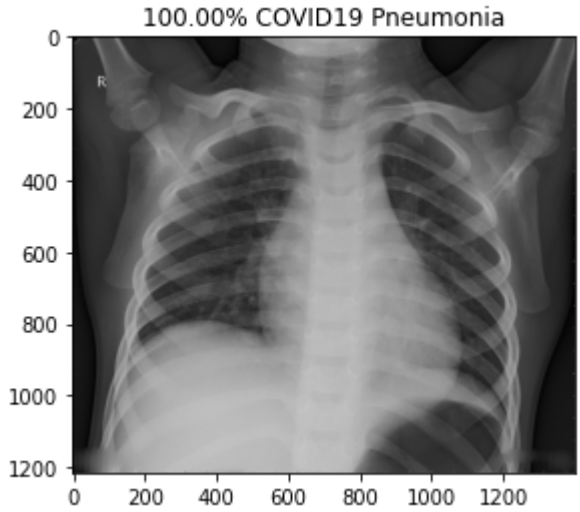
normal/NORMAL2-IM-1419-0001.jpeg



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.

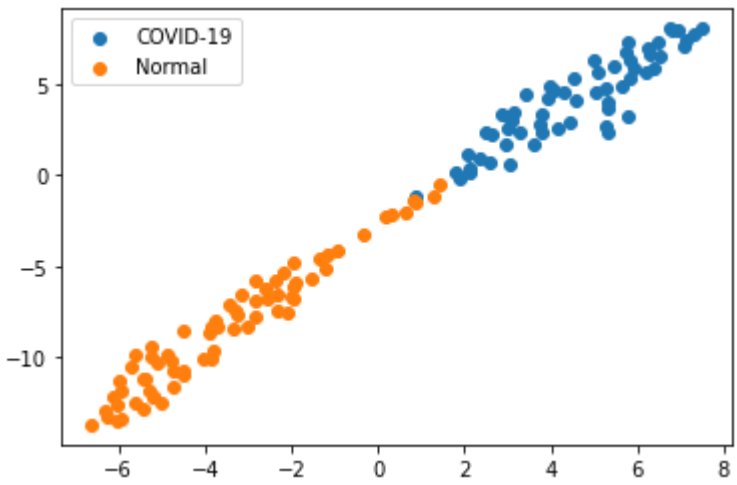
In [ ]:

```
from sklearn.manifold import TSNE
intermediate_layer_model = 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,see
d=42,class_mode="binary")

labels = tsne_data_generator.classes
print(tsne_data_generator.class_indices)

X = TSNE().fit_transform(intermediate_layer_model.predict_generator(tsne_data_ge
nerator, verbose=1))

classes = ["COVID-19", "Normal"]
for i in range(2):
    cluster = X[np.where(labels == i)]
    plt.scatter(cluster[:, 0], cluster[:, 1], label = classes[i])
plt.legend()
```

```
Found 130 images belonging to 2 classes.
{'covid': 0, 'normal': 1}

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:11: Use
rWarning: `Model.predict_generator` is deprecated and will be remove
d in a future version. Please use `Model.predict`, which supports ge
nerators.
  # This is added back by InteractiveShellApp.init_path()


130/130 [==============================] - 4s 29ms/step

/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:78
3: FutureWarning: The default initialization in TSNE will change fro
m 'random' to 'pca' in 1.2.
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:79
3: FutureWarning: The default learning rate in TSNE will change from
200.0 to 'auto' in 1.2.
  FutureWarning,
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7fb2a540c850>
```



In [5]:

```
!jupyter nbconvert --to html '/content/drive/MyDrive/CS 542 Machine Learning/Cha
llenge/Covid_Data_GradientCrescent/task1_template.ipynb'
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/CS 542 Mac
hine Learning/Challenge/Covid_Data_GradientCrescent/task1_template.i
pynb to html
[NbConvertApp] Writing 1685207 bytes to /content/drive/MyDrive/CS 54
2 Machine Learning/Challenge/Covid_Data_GradientCrescent/task1_templ
ate.html
```