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 [3]:
```
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'
print(tf.test.gpu_device_name())
print(tf.__version__)
```

```
/device:GPU:0
2.4.1
```

Load Image Data

In [4]:
```
DATA_LIST = os.listdir('/content/drive/MyDrive/cs542/two/train')
DATASET_PATH  = '/content/drive/MyDrive/cs542/two/train'
TEST_DIR =  '/content/drive/MyDrive/cs542/two/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 25
NUM_EPOCHS    = 90
LEARNING_RATE = 0.0001
NUM_FREEZE = 100
```

Generate Training and Validation Batches

In [18]:
```
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")
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data_generator.py:342: UserWarning: Thi
s 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.

In [33]:
```
pretrained = tf.keras.applications.InceptionV3(
    include_top=False,
    weights="imagenet",
    input_shape=(224, 224, 3),
    classes=2
)

preprocess_input = tf.keras.applications.inception_v3.preprocess_input

print(f"Number of layers in the pretrained model: {len(pretrained.layers)}")
for i in range(NUM_FREEZE):
    pretrained.layers[i].trainable = False
```

```
Number of layers in the pretrained model: 311
```

In [34]:
```
training_layers = tf.keras.Sequential([
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.05)
])

prediction_layer = tf.keras.layers.Dense(1, activation="sigmoid")
```

In [35]:
```
inputs = tf.keras.Input(shape=(None, None, 3))
x = preprocess_input(inputs)
x = pretrained(x)
x = training_layers(x)
dense_feature = tf.keras.layers.Dense(32, activation='relu')
x = dense_feature(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

In [22]:
```
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_4 (InputLayer)         [(None, None, None, 3)]   0         
_________________________________________________________________
tf.math.truediv_1 (TFOpLambd (None, None, None, 3)     0         
_________________________________________________________________
tf.math.subtract_1 (TFOpLamb (None, None, None, 3)     0         
_________________________________________________________________
inception_v3 (Functional)    (None, 5, 5, 2048)        21802784  
_________________________________________________________________
sequential_1 (Sequential)    (None, 128)               262272    
_________________________________________________________________
dense_5 (Dense)              (None, 32)                4128      
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33        
=================================================================
Total params: 22,069,217
Trainable params: 19,892,801
Non-trainable params: 2,176,416
_________________________________________________________________
```

In [36]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(lr=LEARNING_RATE),
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [37]:
```
checkpoint_filepath = '/content/tmp/checkpoint/'
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    sav_freq = 'epoch',
    save_weights_only=True,
    monitor='loss',
    mode='min',
    save_best_only=True)
```

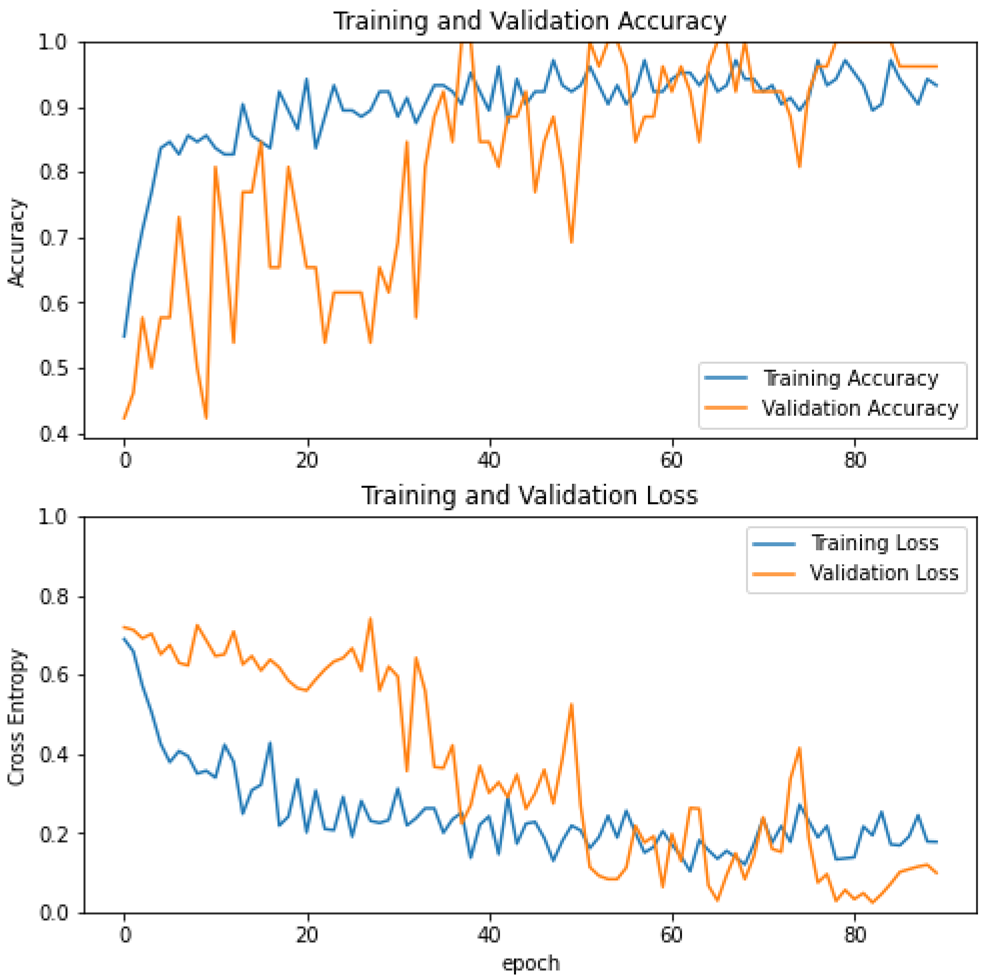
[5 points] Train Model

In [38]:
```
#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(train_batches,
                    epochs=NUM_EPOCHS,
                    validation_data=valid_batches,
                    callbacks = [model_checkpoint_callback])
```

```
Epoch 71/90
5/5 [==============================] - 5s 1s/step - loss: 0.3082 - accuracy: 0.9004 - val_loss: 0.2396 - val
_accuracy: 0.9231
Epoch 72/90
5/5 [==============================] - 5s 898ms/step - loss: 0.1579 - accuracy: 0.9415 - val_loss: 0.1595 -
val_accuracy: 0.9231
Epoch 73/90
5/5 [==============================] - 5s 855ms/step - loss: 0.1705 - accuracy: 0.9185 - val_loss: 0.1521 -
val_accuracy: 0.9231
Epoch 74/90
5/5 [==============================] - 5s 1s/step - loss: 0.1568 - accuracy: 0.9414 - val_loss: 0.3368 - val
_accuracy: 0.8846
Epoch 75/90
5/5 [==============================] - 5s 900ms/step - loss: 0.2359 - accuracy: 0.8997 - val_loss: 0.4148 -
val_accuracy: 0.8077
Epoch 76/90
5/5 [==============================] - 5s 939ms/step - loss: 0.2216 - accuracy: 0.9197 - val_loss: 0.1873 -
val_accuracy: 0.9231
Epoch 77/90
5/5 [==============================] - 5s 911ms/step - loss: 0.2456 - accuracy: 0.9554 - val_loss: 0.0745 -
```

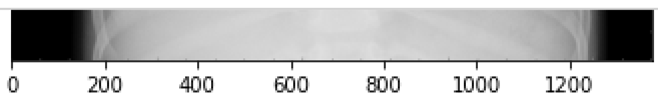
[5 points] Plot Accuracy and Loss During Training

In [39]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```

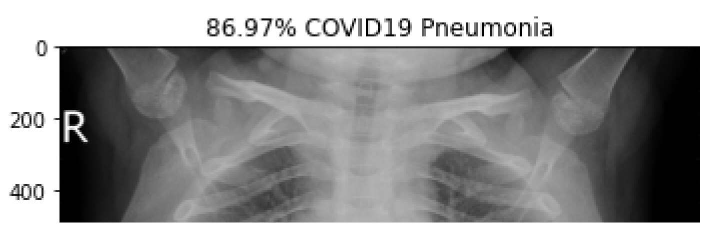


Plot Test Results

In [40]:
```
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)

    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()
```




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



In [44]:
```
x = model.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
                             use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
 3/18 [====>.........................] - ETA: 0s - loss: 0.1138 - accuracy: 1.0000

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.ev
aluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which su
pports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '

18/18 [==============================] - 1s 36ms/step - loss: 0.1333 - accuracy: 1.0000
Test loss: 0.13325661420822144
Test accuracy: 1.0
```

[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 [42]:
```
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.models.Model(inputs=model.input,
                                                 outputs=model.layers[-2].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)
y = tsne.fit_transform(intermediate_output)
```

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

In [43]:
```
plt.scatter(y[:,0],y[:,1], c = tsne_data_generator.labels)
```

Out[43]: <matplotlib.collections.PathCollection at 0x7f4636bfb350>