Daniel Delijani

CS 542

Professor Bargal

Final Report

Model Writeup – Lung Image Classification

Task 1 – COVID, Normal

Part 1: Architecture Description

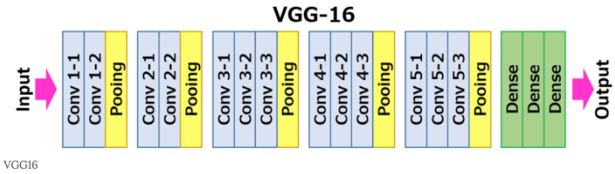


Figure 1

The model begins with using VGG16 as a base model. This model consists of 19 layers, as shown in figure 1. After careful experimentation, I found that providing the greatest validation accuracy when the last 5 layers were unfrozen, while the first 14 layers' weights remain the same as the weights the model generates after being tested on ImageNet. This ultimately results in a base model that contains 7,635,264 non-trainable parameters and 7,079,424 trainable parameters, giving the base model a total of 14,714,688 parameters. After this, I flatten the output and I decided to implement a fully connected layer with 256 neurons with relu as an activation function. I found 256 to be the perfect amount of neurons for the final hidden layer, and relu to be the activation function that led to the greatest validation accuracy. Then, to prevent overfitting, I added a dropout layer with a dropout rate of .25. This I found to be perfect in allowing the validation accuracy to increase right alongside the train accuracy. Additionally, after this layer I create the output layer as a fully connected layer with one output neuron (for binary classification), and sigmoid activation to output probabilities. This gave the model as a whole 21,137,729 total parameters, 13,502,465 of them

trainable and 7,635,264 of them non-trainable. I ended up using a very small learning rate (0.00001), as the model tended to reach a very high accuracy very quickly, so there was no need to use a high learning rate as this would allow the weights to precisely reach their respective local minima. For the loss function, I utilized binary cross entropy as it is both logically fitting and experimentally effective. For my optimizer, I used RMSProp. I noticed that it converged on 1.0 accuracy and 1.0 validation accuracy rather quickly; thus, to ensure that the model kept improving to a certain extent, I utilized RMSProp to keep an adaptive learning rate that would ensure that the gradient would not vanish, keeping each gradient descent update effective.

Part 2: Plot and comment on the accuracy and the loss

As you can tell from Figure 2, both the test accuracy and the test loss for the deep network were incredibly effective very quickly. The train and validation accuracy for the model essentially was 1.0 for almost the entire time, while the train and validation loss was around 0 the entire time as well. This, as mentioned in the previous section ultimately contributed to various decisions I made throughout the development of the model architecture.

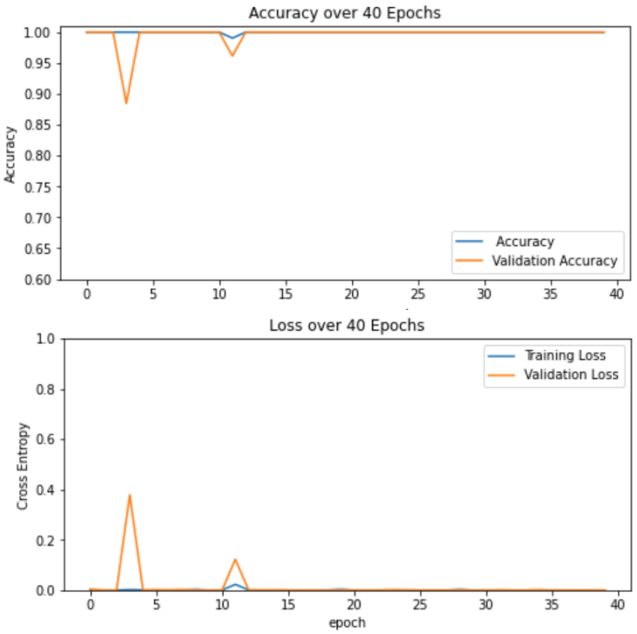


Figure 2

Part 3: Plot and comment on the t-SNE visualizations

I decided to take outputs of the flatten layer, which is the fourth to last layer and contains 25088 neurons. I felt that this layer with its many features would be able to well-distinguish the two classes, especially because there were 5 layers previous to it that have weights that have been optimized for this dataset. Ultimately, we get a graph that clearly distinguishes the two classes with the exception of one point as you can see in figure 3, so it ultimately did a fantastic job of developing meaningful and unique features for the images.

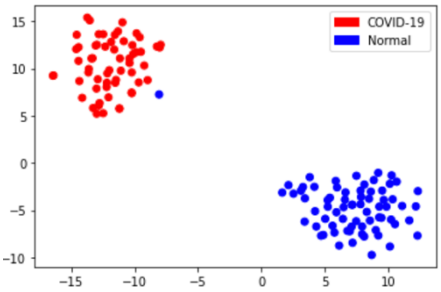


Figure 3

Part 4: Bonus

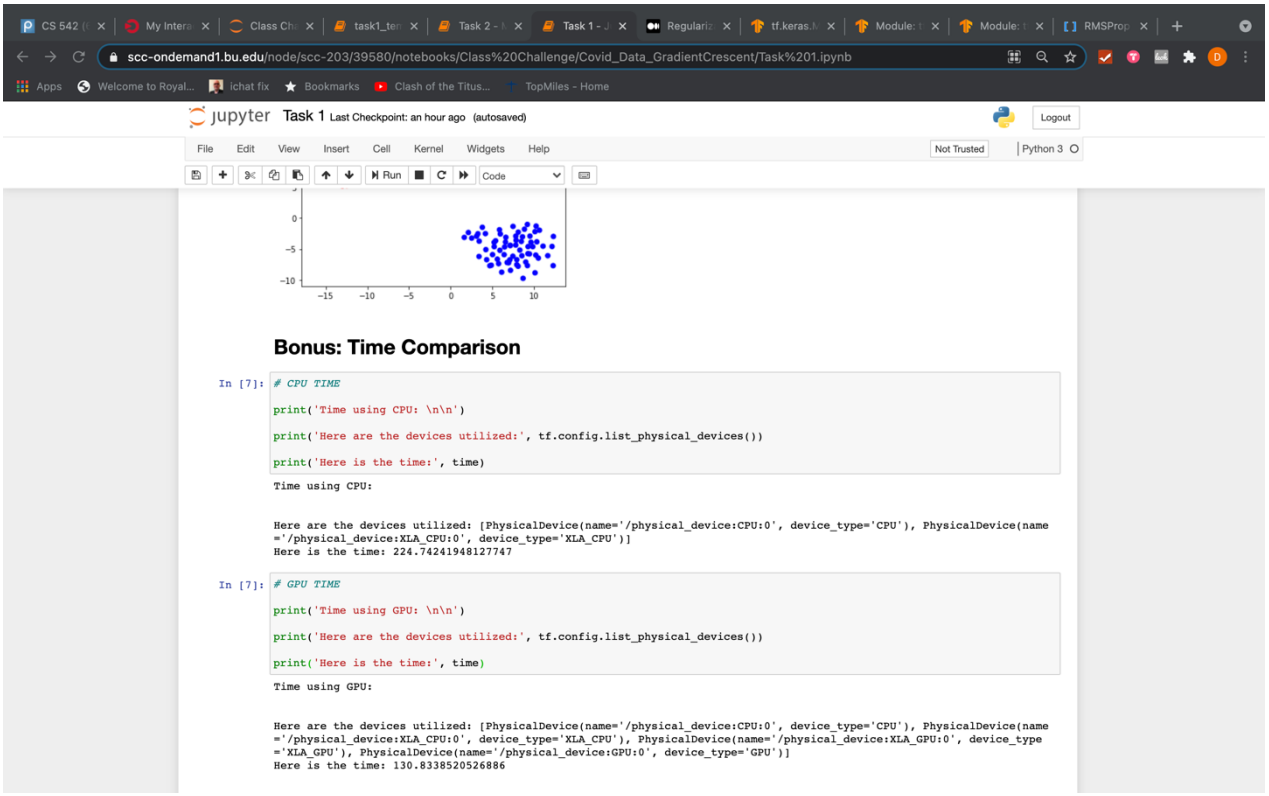


Figure 4

SCC cluster. I used the SCC cluster to launch Jupyter Notebook as you can see from the URL, and at the top of the notebook I included the line

os.environ['CUDA_VISIBLE_DEVICES'] = '-1' and fit the model alongside time.time() to receive the training time for the CPU. Then, I restarted the kernel and did the same for os.environ['CUDA_VISIBLE_DEVICES'] = '0'. Ultimately, the results are as follows:

CPU training time: 224.74241948127747

GPU training time: 130.8338520526886 seconds

A detailed description of what hardware tensorflow had access to for each respective training cycle can be seen in the output of the code in figure 4.

Task 2 – COVID, Normal, Bacterial Pneumonia, Viral Pneumonia

Part 1: Architecture Description + Comparison

For my first and primary model for task 2, the main problem I encountered throughout its development was overfitting. I had many models utilizing VGG16 that would obtain 1.0 train accuracy and test accuracy of about .65, so many of my architectural decisions are centered around this. Firstly, I use pretrained model VGG16 as a base model, with weights initialized to those when optimized on ImageNet (See Figure 1 for details on the model). This is because trying to use other base models simply did not produce good results, as I will show in my experiment with Task 2, Model 2. My first decision that helped prevent overfitting was to freeze 17 of the 19 layers of the base model. This gave the model far less parameters, which ultimately allowed it not overfit. The base model thus had 2,359,808 trainable parameters with the last two layers unfrozen, with 12,354,880 untrainable parameters throughout the first 17 layers for a total of 14,714,688 parameters. Next, I implement a flatten layer. Then, I create a dropout layer with a value of .4 to prevent overfitting. Then, I have a fully connected layer with 256 output neurons, with activation relu, and regularization function l2 norm with l = .3 to

further help prevent overfitting. Then I use yet another dropout layer with a rate of .4. Finally, for my output layer I have a fully connected layer with 4 output neurons (4 classes), with activation softmax to output probabilities, and regularization function l2 norm with l = .3 once again to help prevent overfitting. This ultimately gave the model as a whole 21,138,500 parameters, 8,783,620 of them trainable and 12,354,880 non-trainable. For the learning rate I went with a relatively small learning rate of .00001 as it had no problem reaching a local minima generally in a short amount of epochs, so this would allow it to narrow in on the perfect minima. I used RMSProp once again for this task as the variable learning rate ultimately allowed the model to fit the data very well. Next, for the loss function I utilize the categorical cross entropy loss function, which had the best results as I expected. Another decision I made was to prevent overfitting in this model was to utilize early stopping by only running 40 epochs. I found this to be the perfect amount where the data was not being overfit, and validation accuracy was highest.

For the second model, I used Resnet 50 as a base model as I read very positive affirmations of it online. I decided to freeze 171 of the 175 layers, leaving the last 4 layers open to training. This gave the base model 4096 trainable parameters and 23,583,616 non-trainable parameters for a total of 23,587,712 parameters. Next in the model is a flatten layer. Then I create a fully connected layer with 256 neurons (which in the past I have realized is very effective) with an activation of relu to help the model learn more efficiently. Then, for the output layer I implement a fully connected layer with 4 output neurons (because 4 categories) with an activation of softmax so it outputs probabilities. For the learning rate, I use a somewhat high rate at .0001 as the model

had trouble learning quickly with lower learning rates. For optimizer function, I use RMSProp again as it was consistently the most effective function. Additionally, I use categorical cross entropy as it worked the best and is a logical use case as well.

In terms of performance, model 1 performed far better than model 2. As you can see in Figure 5 and Figure 6, the accuracy for model 1 is far better than that for model 2. Ultimately, the test accuracy on the unseen data for model 1 is .75, whereas for model 2 it is .27. I attribute this difference to be the result of the base model. I think that for this classification task, VGG16 as a base model proves to be far more effective, and that though it may be possible to tweak parameters to achieve a decent model with Resnet 50, it is pointless as the same modifications applied to a model with VGG16 as its base model would prove far more effective. Additionally, to comment on model one specifically, all the measures I took to avoid overfitting proved successful as the validation accuracy is right on par with the train accuracy throughout the model fitting.

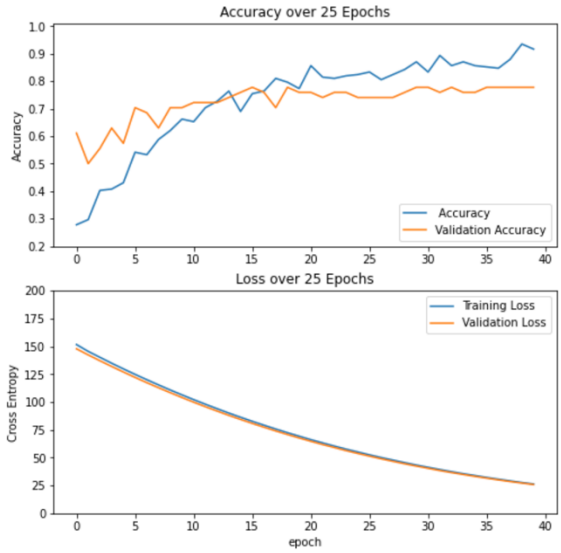


Figure 6 – Model 1 Metrics

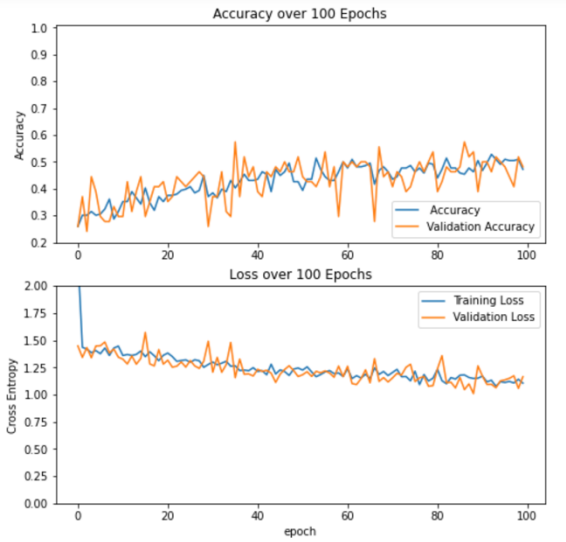


Figure 5 – Model 2 Metrics

Part 3: Plot and comment on the t-SNE visualizations

For both models, I found the first dense layer to have the best features that best separated the data into distinct clusters. For model 1, there are essentially 3 separate

clusters with one cluster that is vaguely separable into two sub-clusters as you can see in figure 7. It did a fantastic job of distinguishing between normal and COVID-19, but the features between bacterial Pnemonia and Viral Pnemonia are very similar, which is largely to be expected as they are similar diseases. For model 2, it seems to have only really separated normal from the rest of the classifications as a result of the Resnet 50 base model that simply was not suited for this task.

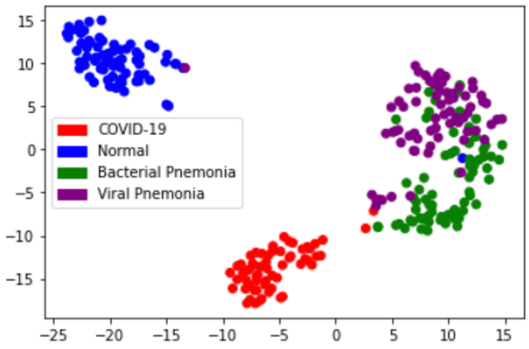


Figure 7 – Model 1 TSNE

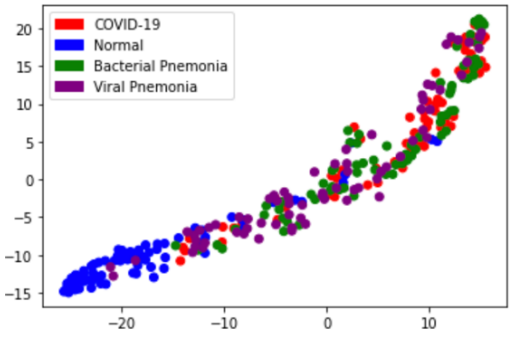


Figure 8 – Model 2 TSNE

This concludes the process of developing the models, from architecture to analysis and visualizations. Overall, though it was much work, it was very rewarding and very engaging! Great way to utilize many of the skills we have been learning in class. Thank you for your time!

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 [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 [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
NUM_EPOCHS    = 40
LEARNING_RATE = 0.0005 # start off with high rate first 0.001 and experiment wit
```

Generate Training and Validation Batches

In [15]:
```
"""train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewi
                                      featurewise_std_normalization = True,width_sh
                                      height_shift_range=0.2,shear_range=0.25,zoom_
                                       zca_whitening = True,channel_shift_range = 2
                                      horizontal_flip = True,vertical_flip = True,
                                      validation_split = 0.2,fill_mode='constant')"

train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50, validation_

train_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE
                                                  shuffle=True,batch_size=BATCH_
                                                  subset = "training",seed=42,
                                                  class_mode="binary")

valid_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE
                                                  shuffle=True,batch_size=BATCH_
                                                  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 [16]:
```
IMG_SHAPE = IMAGE_SIZE + (3,)
base_model = tf.keras.applications.VGG16(input_shape=IMG_SHAPE,
                                         include_top=False,
                                         weights='imagenet')
base_model.trainable = True

print("Number of layers in the base model: ", len(base_model.layers))

# Fine-tune from this layer onwards
fine_tune_at = 15
```

```
# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable =  False

base_model.summary()
```

```
Number of layers in the base model:  19
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (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
=================================================================
Total params: 14,714,688
Trainable params: 7,079,424
Non-trainable params: 7,635,264
_________________________________________________________________
```

In [21]:

```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = base_model(inputs, training=False)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(256, activation = 'relu')(x)
x = tf.keras.layers.Dropout(.25)(x)
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(inputs, outputs)
```

```
model.summary()
```

```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_3 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               6422784
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,137,729
Trainable params: 13,502,465
Non-trainable params: 7,635,264
_________________________________________________________________
```

[5 points] Train Model

In [29]:

```
import time

learning_rate = 0.00001

model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
              optimizer = tf.keras.optimizers.RMSprop(lr=learning_rate),
              metrics=['accuracy'])

starttime = time.time()

history = model.fit(train_batches,
                    epochs=40,
                    validation_data=valid_batches)

endtime = time.time()

time = endtime - starttime
```

```
Epoch 1/40
11/11 [==============================] - 6s 524ms/step - loss: 2.2270e-04 - accu
racy: 1.0000 - val_loss: 0.0040 - val_accuracy: 1.0000
Epoch 2/40
11/11 [==============================] - 5s 456ms/step - loss: 2.9803e-05 - accu
racy: 1.0000 - val_loss: 2.0051e-04 - val_accuracy: 1.0000
Epoch 3/40
11/11 [==============================] - 5s 452ms/step - loss: 1.3409e-05 - accu
racy: 1.0000 - val_loss: 1.6051e-04 - val_accuracy: 1.0000
Epoch 4/40
11/11 [==============================] - 5s 491ms/step - loss: 0.0018 - accurac
y: 1.0000 - val_loss: 0.3787 - val_accuracy: 0.8846
Epoch 5/40
11/11 [==============================] - 5s 484ms/step - loss: 5.1076e-04 - accu
racy: 1.0000 - val_loss: 4.8395e-04 - val_accuracy: 1.0000
Epoch 6/40
11/11 [==============================] - 5s 475ms/step - loss: 9.1783e-06 - accu
racy: 1.0000 - val_loss: 0.0017 - val_accuracy: 1.0000
```

```
Epoch 7/40
11/11 [==============================] - 5s 453ms/step - loss: 1.7729e-04 - accu
racy: 1.0000 - val_loss: 9.7333e-04 - val_accuracy: 1.0000
Epoch 8/40
11/11 [==============================] - 5s 472ms/step - loss: 1.3464e-04 - accu
racy: 1.0000 - val_loss: 0.0021 - val_accuracy: 1.0000
Epoch 9/40
11/11 [==============================] - 5s 472ms/step - loss: 0.0026 - accurac
y: 1.0000 - val_loss: 4.1974e-05 - val_accuracy: 1.0000
Epoch 10/40
11/11 [==============================] - 5s 468ms/step - loss: 1.2027e-04 - accu
racy: 1.0000 - val_loss: 1.4081e-04 - val_accuracy: 1.0000
Epoch 11/40
11/11 [==============================] - 5s 476ms/step - loss: 2.6605e-05 - accu
racy: 1.0000 - val_loss: 6.4005e-05 - val_accuracy: 1.0000
Epoch 12/40
11/11 [==============================] - 5s 474ms/step - loss: 0.0229 - accurac
y: 0.9904 - val_loss: 0.1224 - val_accuracy: 0.9615
Epoch 13/40
11/11 [==============================] - 5s 468ms/step - loss: 7.4647e-04 - accu
racy: 1.0000 - val_loss: 0.0012 - val_accuracy: 1.0000
Epoch 14/40
11/11 [==============================] - 5s 474ms/step - loss: 3.2215e-05 - accu
racy: 1.0000 - val_loss: 5.9854e-04 - val_accuracy: 1.0000
Epoch 15/40
11/11 [==============================] - 5s 469ms/step - loss: 8.2588e-05 - accu
racy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 16/40
11/11 [==============================] - 5s 496ms/step - loss: 4.0852e-05 - accu
racy: 1.0000 - val_loss: 3.7106e-04 - val_accuracy: 1.0000
Epoch 17/40
11/11 [==============================] - 5s 473ms/step - loss: 9.2613e-05 - accu
racy: 1.0000 - val_loss: 1.6079e-04 - val_accuracy: 1.0000
Epoch 18/40
11/11 [==============================] - 5s 498ms/step - loss: 2.7357e-05 - accu
racy: 1.0000 - val_loss: 2.4570e-04 - val_accuracy: 1.0000
Epoch 19/40
11/11 [==============================] - 5s 465ms/step - loss: 5.4286e-04 - accu
racy: 1.0000 - val_loss: 1.0493e-05 - val_accuracy: 1.0000
Epoch 20/40
11/11 [==============================] - 5s 459ms/step - loss: 0.0031 - accurac
y: 1.0000 - val_loss: 2.7993e-04 - val_accuracy: 1.0000
Epoch 21/40
11/11 [==============================] - 5s 462ms/step - loss: 1.1187e-05 - accu
racy: 1.0000 - val_loss: 3.5999e-04 - val_accuracy: 1.0000
Epoch 22/40
11/11 [==============================] - 5s 494ms/step - loss: 5.4345e-06 - accu
racy: 1.0000 - val_loss: 3.8878e-05 - val_accuracy: 1.0000
Epoch 23/40
11/11 [==============================] - 5s 459ms/step - loss: 2.5656e-06 - accu
racy: 1.0000 - val_loss: 6.7928e-04 - val_accuracy: 1.0000
Epoch 24/40
11/11 [==============================] - 5s 476ms/step - loss: 1.0248e-05 - accu
racy: 1.0000 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 25/40
11/11 [==============================] - 5s 489ms/step - loss: 1.2623e-05 - accu
racy: 1.0000 - val_loss: 8.9979e-04 - val_accuracy: 1.0000
Epoch 26/40
11/11 [==============================] - 5s 450ms/step - loss: 3.5261e-06 - accu
racy: 1.0000 - val_loss: 3.7036e-04 - val_accuracy: 1.0000
Epoch 27/40
11/11 [==============================] - 5s 464ms/step - loss: 1.7266e-06 - accu
racy: 1.0000 - val_loss: 6.5534e-05 - val_accuracy: 1.0000
Epoch 28/40
11/11 [==============================] - 5s 467ms/step - loss: 8.9506e-06 - accu
```

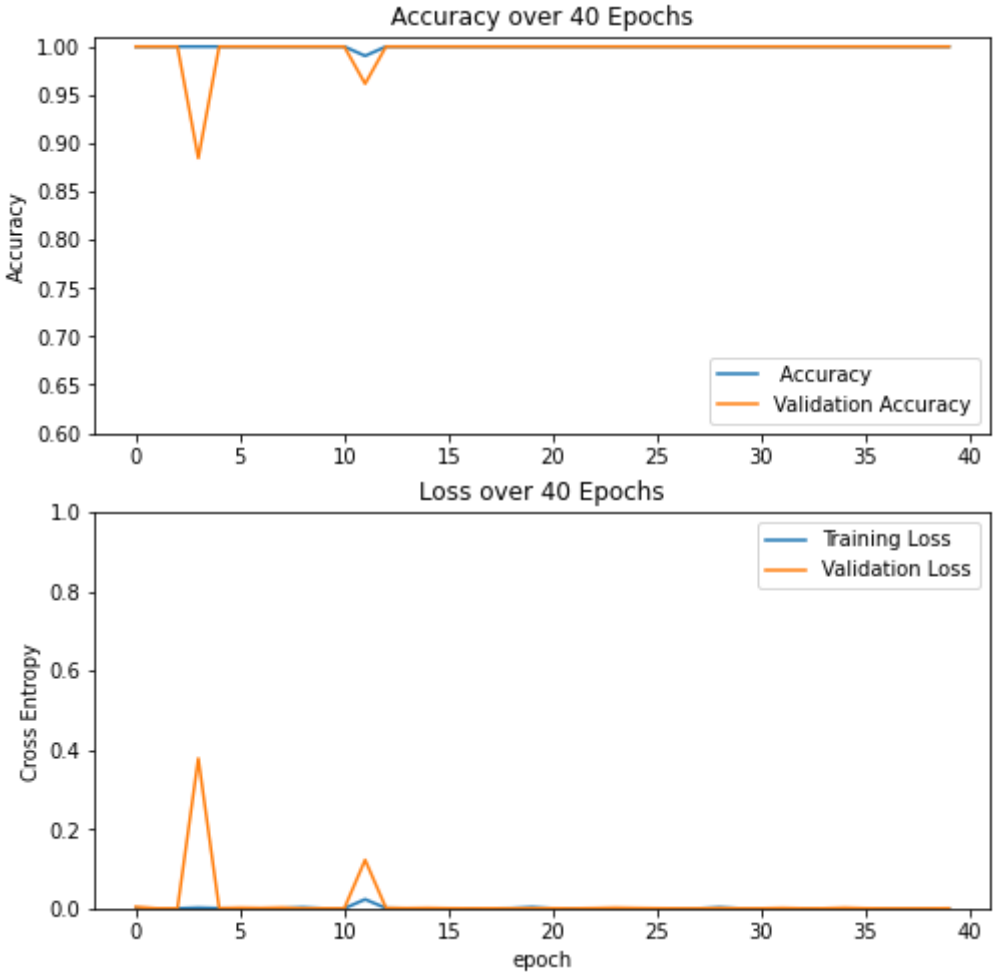
```
racy: 1.0000 - val_loss: 3.3013e-06 - val_accuracy: 1.0000
Epoch 29/40
11/11 [==============================] - 5s 477ms/step - loss: 0.0026 - accurac
y: 1.0000 - val_loss: 7.3713e-04 - val_accuracy: 1.0000
Epoch 30/40
11/11 [==============================] - 5s 473ms/step - loss: 6.1710e-06 - accu
racy: 1.0000 - val_loss: 6.3861e-05 - val_accuracy: 1.0000
Epoch 31/40
11/11 [==============================] - 5s 471ms/step - loss: 5.6818e-06 - accu
racy: 1.0000 - val_loss: 1.3577e-04 - val_accuracy: 1.0000
Epoch 32/40
11/11 [==============================] - 5s 480ms/step - loss: 6.4407e-06 - accu
racy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 33/40
11/11 [==============================] - 5s 465ms/step - loss: 3.3217e-06 - accu
racy: 1.0000 - val_loss: 1.0078e-04 - val_accuracy: 1.0000
Epoch 34/40
11/11 [==============================] - 5s 454ms/step - loss: 1.3515e-06 - accu
racy: 1.0000 - val_loss: 2.5246e-04 - val_accuracy: 1.0000
Epoch 35/40
11/11 [==============================] - 5s 481ms/step - loss: 3.0807e-06 - accu
racy: 1.0000 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 36/40
11/11 [==============================] - 5s 483ms/step - loss: 2.0439e-06 - accu
racy: 1.0000 - val_loss: 3.0789e-04 - val_accuracy: 1.0000
Epoch 37/40
11/11 [==============================] - 5s 463ms/step - loss: 3.3099e-06 - accu
racy: 1.0000 - val_loss: 8.0719e-06 - val_accuracy: 1.0000
Epoch 38/40
11/11 [==============================] - 5s 481ms/step - loss: 3.5043e-05 - accu
racy: 1.0000 - val_loss: 5.6937e-07 - val_accuracy: 1.0000
Epoch 39/40
11/11 [==============================] - 5s 481ms/step - loss: 1.6826e-06 - accu
racy: 1.0000 - val_loss: 6.4652e-05 - val_accuracy: 1.0000
Epoch 40/40
11/11 [==============================] - 5s 492ms/step - loss: 5.5023e-07 - accu
racy: 1.0000 - val_loss: 3.9884e-04 - val_accuracy: 1.0000
```

[5 points] Plot Accuracy and Loss During Training

In [30]:

```
import matplotlib.pyplot as plt

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=' Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([.6,1.01])
plt.title('Accuracy over 40 Epochs')

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('Loss over 40 Epochs')
plt.xlabel('epoch')
plt.show()
```



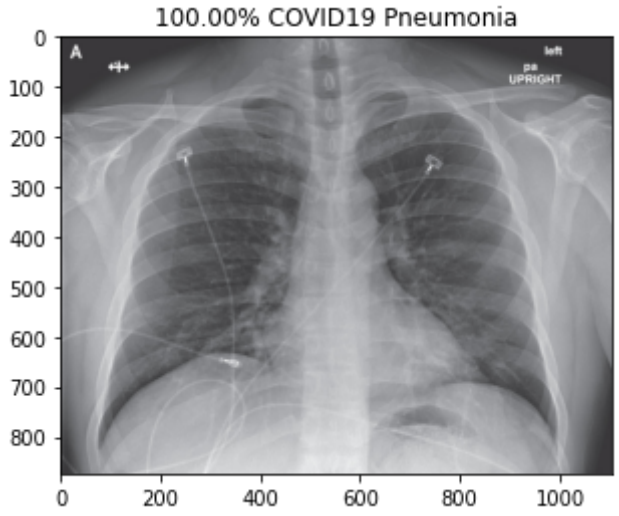
Plot Test Results

In [31]:

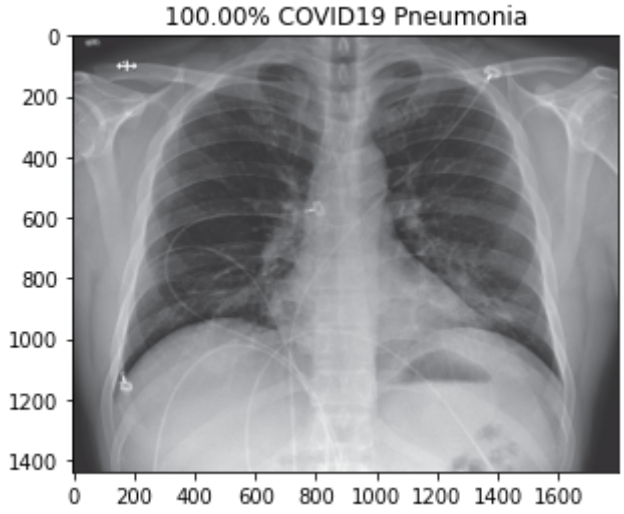
```
import matplotlib.image as mpimg

test_datagen = ImageDataGenerator(rescale=1. / 255)
eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZ
                                                  batch_size=1,shuffle=True,seed
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()
```

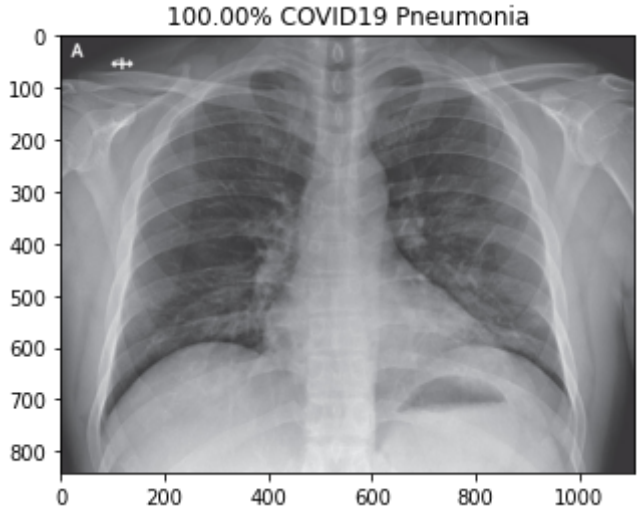
```
Found 18 images belonging to 2 classes.
18/18 [==============================] - 1s 56ms/step
covid/nejmoa2001191_f3-PA.jpeg
```



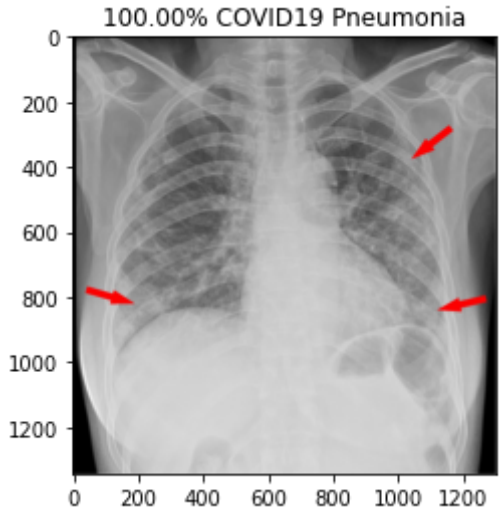

```
covid/nejmoa2001191_f4.jpeg
```



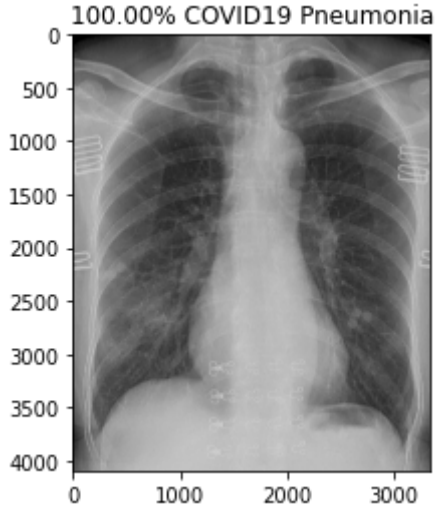

```
covid/nejmoa2001191_f5-PA.jpeg
```



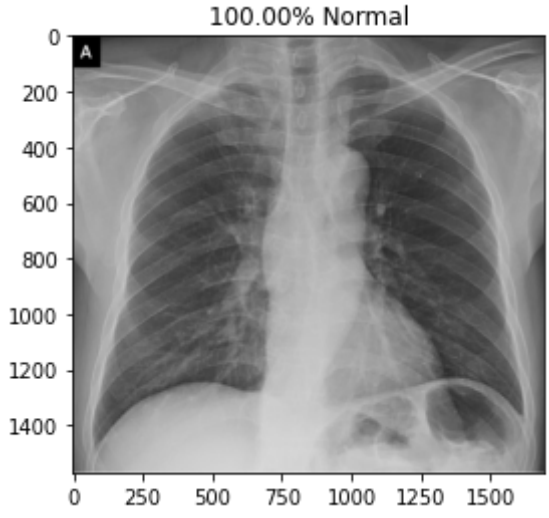

```
covid/radiol.2020200490.fig3.jpeg
```



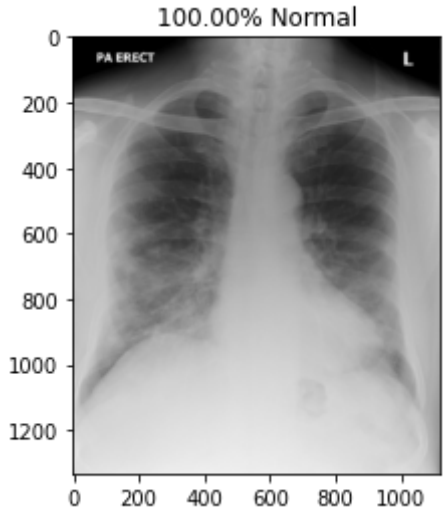
covid/ryct.2020200028.fig1a.jpeg



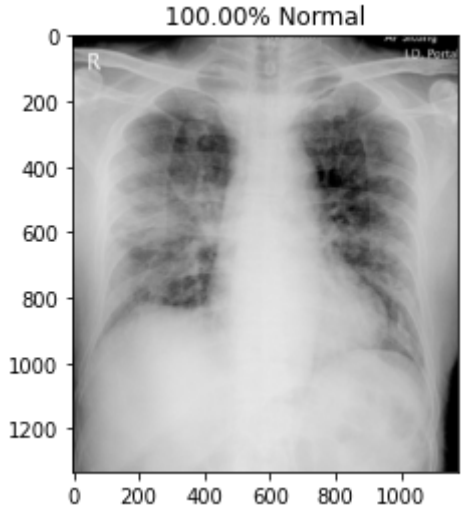
covid/ryct.2020200034.fig2.jpeg



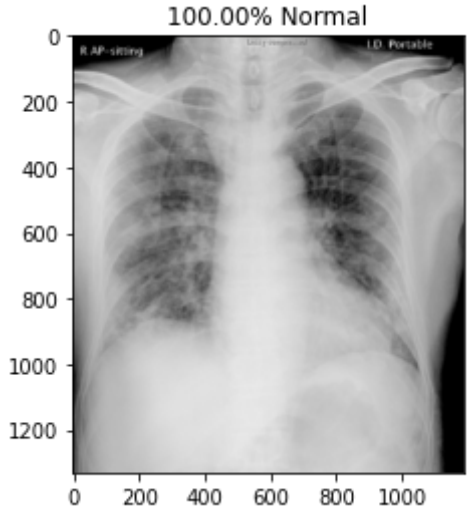
covid/ryct.2020200034.fig5-day0.jpeg



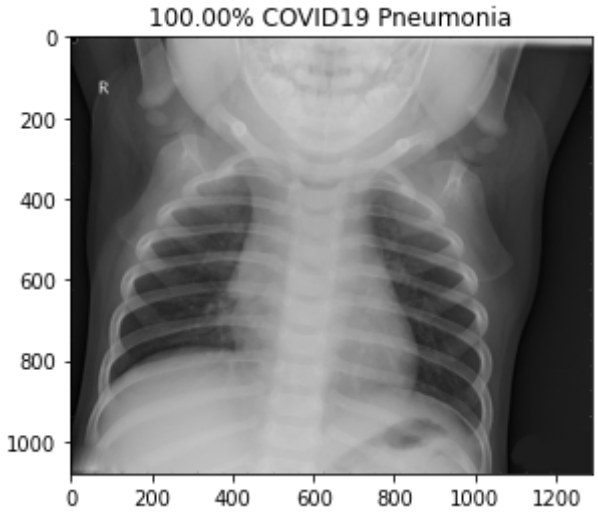
covid/ryct.2020200034.fig5-day4.jpeg



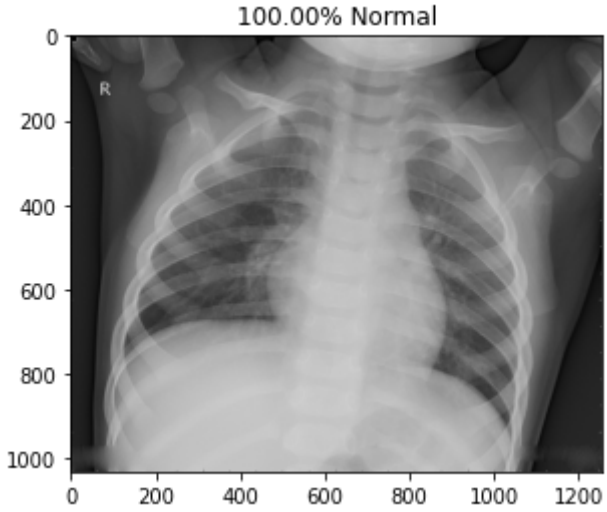
covid/ryct.2020200034.fig5-day7.jpeg



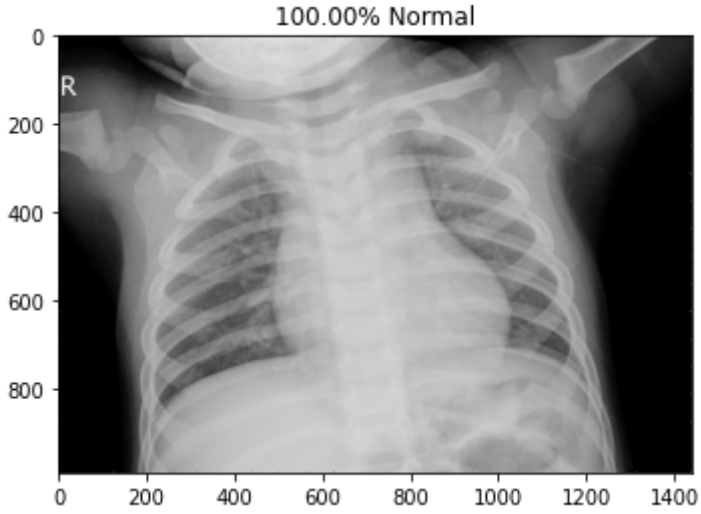
normal/NORMAL2-IM-1385-0001.jpeg



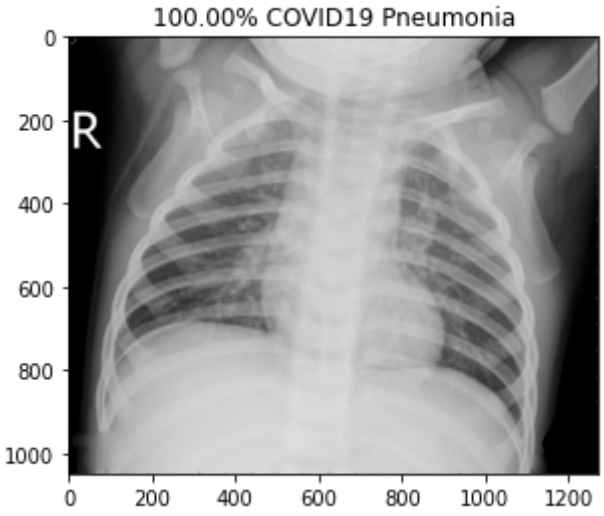
normal/NORMAL2-IM-1396-0001.jpeg



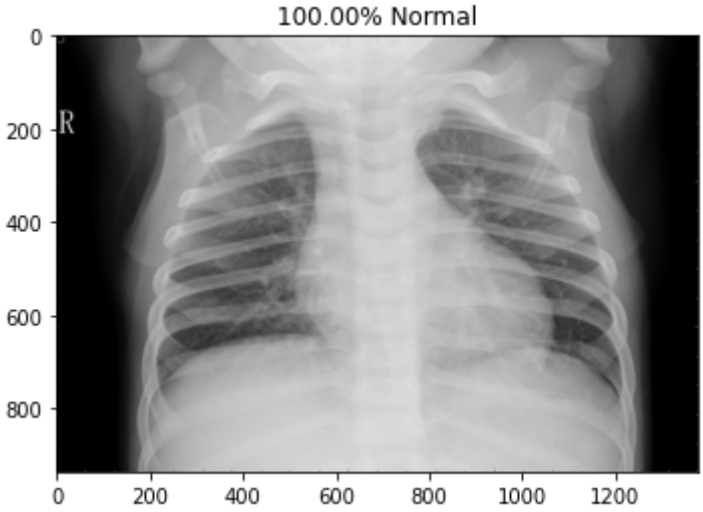
normal/NORMAL2-IM-1400-0001.jpeg



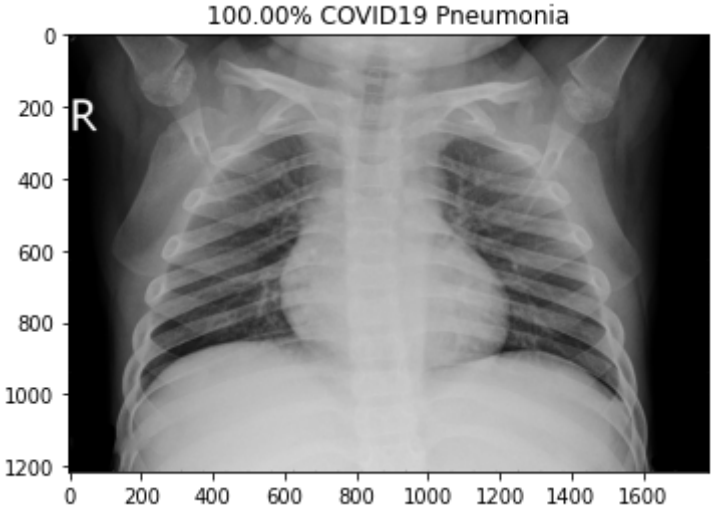
normal/NORMAL2-IM-1401-0001.jpeg



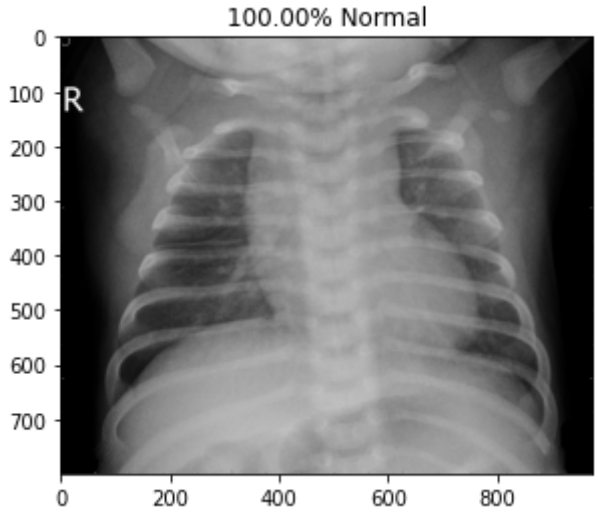
normal/NORMAL2-IM-1406-0001.jpeg



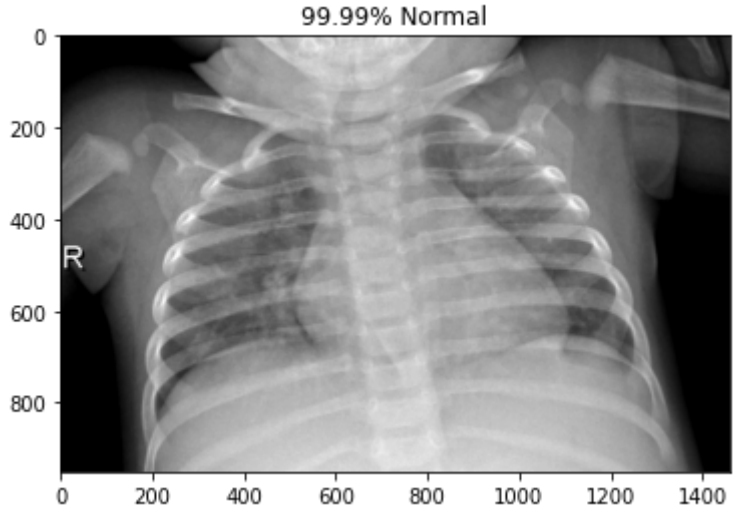
normal/NORMAL2-IM-1412-0001.jpeg



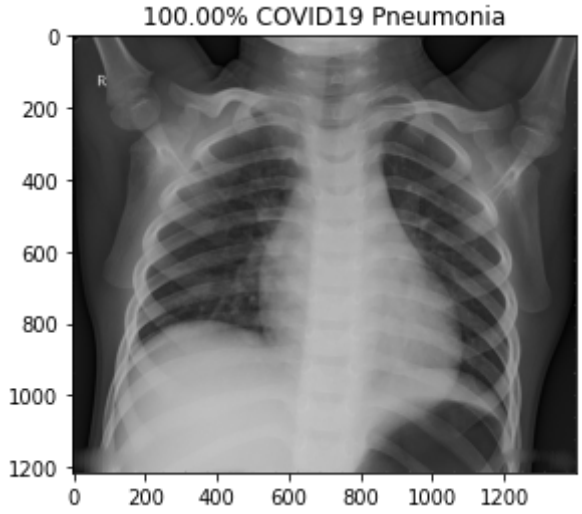
normal/NORMAL2-IM-1419-0001.jpeg



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 [32]:
```
from tensorflow.keras import models
from sklearn.manifold import TSNE
import matplotlib.patches as mpatches
```

Creating the Reduced Data and Classifications

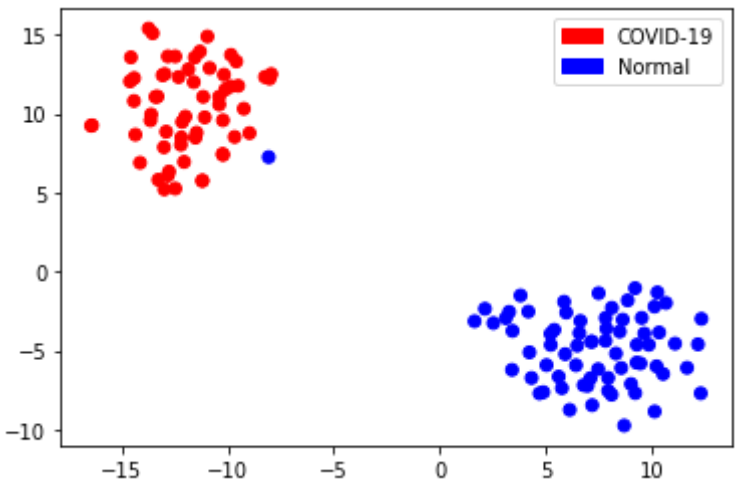
In [33]:
```
intermediate_layer_model = models.Model(inputs=model.input,
                                        outputs=model.get_layer('flatten_3').out

tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=
                                                       batch_size=1,shuffle=False,see

intermediate_output = intermediate_layer_model.predict(tsne_data_generator)

reducedData = TSNE(n_components=2).fit_transform(intermediate_output)

class_array = tsne_data_generator.labels

tsne_data_generator.class_indices
```

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

Out[33]:
```
{'covid': 0, 'normal': 1}
```

Plotting

In [34]:
```
colormapping = {0: 'red', 1: 'blue'}
colors = [colormapping[i] for i in class_array]

plt.scatter(x = reducedData[:, 0], y = reducedData[:, 1], c = colors)

red_patch = mpatches.Patch(color='red', label='COVID-19')
blue_patch = mpatches.Patch(color='blue', label='Normal')
plt.legend(handles=[red_patch, blue_patch])
plt.show()
```



Bonus: Time Comparison

In [7]:
```
# CPU TIME

print('Time using CPU: \n\n')

print('Here are the devices utilized:', tf.config.list_physical_devices())

print('Here is the time:', time)
```

```
Time using CPU:


Here are the devices utilized: [PhysicalDevice(name='/physical_device:CPU:0', de
vice_type='CPU'), PhysicalDevice(name='/physical_device:XLA_CPU:0', device_type
='XLA_CPU')]
Here is the time: 224.74241948127747
```

In [7]:
```
# GPU TIME

print('Time using GPU: \n\n')

print('Here are the devices utilized:', tf.config.list_physical_devices())

print('Here is the time:', time)
```

```
Time using GPU:


Here are the devices utilized: [PhysicalDevice(name='/physical_device:CPU:0', de
vice_type='CPU'), PhysicalDevice(name='/physical_device:XLA_CPU:0', device_type
='XLA_CPU'), PhysicalDevice(name='/physical_device:XLA_GPU:0', device_type='XLA_
GPU'), PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
Here is the time: 130.8338520526886
```

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

Task 2 (Model 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] Multi-class Classification

In [2]:
```
import os

import tensorflow as tf
from tensorflow import keras
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]: '2.3.1'

Load Image Data

In [3]:
```
DATA_LIST = os.listdir('all/train')
DATASET_PATH  = 'all/train'
TEST_DIR =  'all/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 10  # try reducing batch size or freeze more layers if your GPU
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experiment wit
```

Generate Training and Validation Batches

In [4]:
```
train_datagen = ImageDataGenerator(rescale=1./255, validation_split = 0.2)


train_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE
                                                  shuffle=True,batch_size=BATCH_
                                                  subset = "training",seed=42,
                                                  class_mode="categorical")

valid_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE
                                                  shuffle=True,batch_size=BATCH_
                                                  subset = "validation",
                                                  seed=42,class_mode="categorica
```

```
Found 216 images belonging to 4 classes.
Found 54 images belonging to 4 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 [49]:
```
IMG_SHAPE = IMAGE_SIZE + (3,)
base_model = tf.keras.applications.VGG16(input_shape=IMG_SHAPE,
                                                 include_top=False,
                                                 weights='imagenet')
base_model.trainable = True

print("Number of layers in the base model: ", len(base_model.layers))

# Fine-tune from this layer onwards
fine_tune_at = 17

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable =  False

base_model.summary()
```

```
Number of layers in the base model:  19
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_11 (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
=================================================================
Total params: 14,714,688
Trainable params: 2,359,808
Non-trainable params: 12,354,880
_________________________________________________________________
```

In [50]:

```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = base_model(inputs, training=False)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dropout(.4)(x)
x = tf.keras.layers.Dense(256, activation = 'relu', kernel_regularizer=keras.reg
x = tf.keras.layers.Dropout(.4)(x)
outputs = tf.keras.layers.Dense(4, activation='softmax', kernel_regularizer=kera
model = tf.keras.Model(inputs, outputs)

model.summary()
```

```
Model: "functional_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
input_12 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_8 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_8 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               6422784
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_17 (Dense)             (None, 4)                 1028
=================================================================
Total params: 21,138,500
Trainable params: 8,783,620
Non-trainable params: 12,354,880
_________________________________________________________________
```

[5 points] Train Model

In [51]:

```
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(from_logits=False),
              optimizer = tf.keras.optimizers.RMSprop(lr=LEARNING_RATE/10),
              metrics=['accuracy'])

history = model.fit(train_batches,
                    epochs=40,
                    validation_data=valid_batches)
```

```
Epoch 1/40
22/22 [==============================] - 7s 308ms/step - loss: 151.5618 - accura
cy: 0.2778 - val_loss: 147.7494 - val_accuracy: 0.6111
Epoch 2/40
22/22 [==============================] - 6s 274ms/step - loss: 145.4193 - accura
cy: 0.2963 - val_loss: 142.2655 - val_accuracy: 0.5000
Epoch 3/40
22/22 [==============================] - 6s 288ms/step - loss: 140.1026 - accura
cy: 0.4028 - val_loss: 137.0128 - val_accuracy: 0.5556
Epoch 4/40
22/22 [==============================] - 6s 292ms/step - loss: 134.8022 - accura
cy: 0.4074 - val_loss: 131.8941 - val_accuracy: 0.6296
Epoch 5/40
22/22 [==============================] - 6s 282ms/step - loss: 129.7278 - accura
cy: 0.4306 - val_loss: 126.9202 - val_accuracy: 0.5741
Epoch 6/40
22/22 [==============================] - 6s 284ms/step - loss: 124.7632 - accura
cy: 0.5417 - val_loss: 122.0277 - val_accuracy: 0.7037
Epoch 7/40
22/22 [==============================] - 6s 291ms/step - loss: 119.9855 - accura
cy: 0.5324 - val_loss: 117.3285 - val_accuracy: 0.6852
Epoch 8/40
22/22 [==============================] - 6s 286ms/step - loss: 115.3144 - accura
cy: 0.5880 - val_loss: 112.7835 - val_accuracy: 0.6296
Epoch 9/40
22/22 [==============================] - 6s 285ms/step - loss: 110.7586 - accura
cy: 0.6204 - val_loss: 108.3299 - val_accuracy: 0.7037
Epoch 10/40
22/22 [==============================] - 6s 279ms/step - loss: 106.3651 - accura
cy: 0.6620 - val_loss: 103.9964 - val_accuracy: 0.7037
Epoch 11/40
22/22 [==============================] - 6s 284ms/step - loss: 102.0836 - accura
cy: 0.6528 - val_loss: 99.7879 - val_accuracy: 0.7222
```

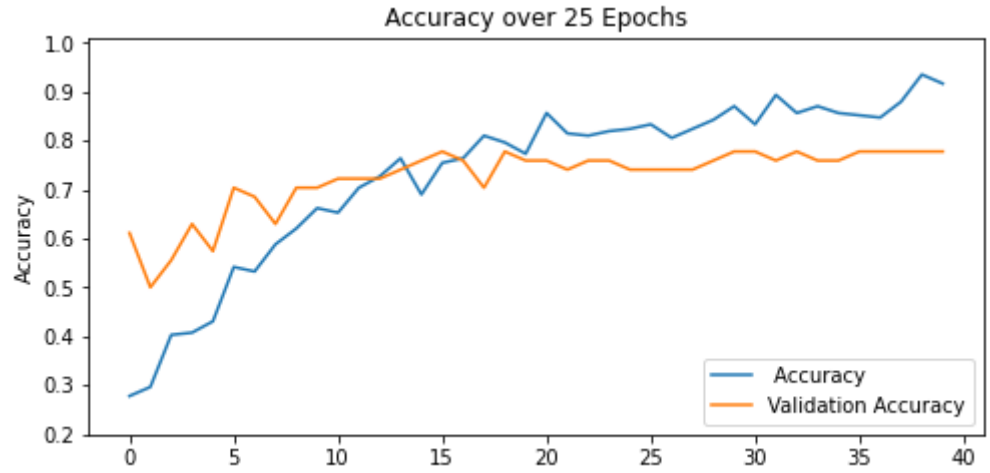
```
Epoch 12/40
22/22 [==============================] - 6s 290ms/step - loss: 97.9428 - accurac
y: 0.7037 - val_loss: 95.7272 - val_accuracy: 0.7222
Epoch 13/40
22/22 [==============================] - 6s 283ms/step - loss: 93.9404 - accurac
y: 0.7269 - val_loss: 91.8075 - val_accuracy: 0.7222
Epoch 14/40
22/22 [==============================] - 6s 281ms/step - loss: 89.9999 - accurac
y: 0.7639 - val_loss: 87.9603 - val_accuracy: 0.7407
Epoch 15/40
22/22 [==============================] - 6s 286ms/step - loss: 86.2617 - accurac
y: 0.6898 - val_loss: 84.3030 - val_accuracy: 0.7593
Epoch 16/40
22/22 [==============================] - 6s 291ms/step - loss: 82.6021 - accurac
y: 0.7546 - val_loss: 80.7351 - val_accuracy: 0.7778
Epoch 17/40
22/22 [==============================] - 6s 282ms/step - loss: 79.0967 - accurac
y: 0.7639 - val_loss: 77.3018 - val_accuracy: 0.7593
Epoch 18/40
22/22 [==============================] - 6s 278ms/step - loss: 75.6758 - accurac
y: 0.8102 - val_loss: 73.9640 - val_accuracy: 0.7037
Epoch 19/40
22/22 [==============================] - 6s 287ms/step - loss: 72.3817 - accurac
y: 0.7963 - val_loss: 70.7609 - val_accuracy: 0.7778
Epoch 20/40
22/22 [==============================] - 6s 278ms/step - loss: 69.2303 - accurac
y: 0.7731 - val_loss: 67.6491 - val_accuracy: 0.7593
Epoch 21/40
22/22 [==============================] - 6s 276ms/step - loss: 66.1221 - accurac
y: 0.8565 - val_loss: 64.6630 - val_accuracy: 0.7593
Epoch 22/40
22/22 [==============================] - 6s 282ms/step - loss: 63.2378 - accurac
y: 0.8148 - val_loss: 61.7728 - val_accuracy: 0.7407
Epoch 23/40
22/22 [==============================] - 6s 290ms/step - loss: 60.3678 - accurac
y: 0.8102 - val_loss: 59.0494 - val_accuracy: 0.7593
Epoch 24/40
22/22 [==============================] - 6s 280ms/step - loss: 57.6199 - accurac
y: 0.8194 - val_loss: 56.3114 - val_accuracy: 0.7593
Epoch 25/40
22/22 [==============================] - 6s 289ms/step - loss: 55.0555 - accurac
y: 0.8241 - val_loss: 53.7656 - val_accuracy: 0.7407
Epoch 26/40
22/22 [==============================] - 6s 288ms/step - loss: 52.4855 - accurac
y: 0.8333 - val_loss: 51.3121 - val_accuracy: 0.7407
Epoch 27/40
22/22 [==============================] - 6s 283ms/step - loss: 50.0622 - accurac
y: 0.8056 - val_loss: 48.9251 - val_accuracy: 0.7407
Epoch 28/40
22/22 [==============================] - 6s 284ms/step - loss: 47.7774 - accurac
y: 0.8241 - val_loss: 46.6579 - val_accuracy: 0.7407
Epoch 29/40
22/22 [==============================] - 6s 292ms/step - loss: 45.4919 - accurac
y: 0.8426 - val_loss: 44.4536 - val_accuracy: 0.7593
Epoch 30/40
22/22 [==============================] - 6s 284ms/step - loss: 43.3128 - accurac
y: 0.8704 - val_loss: 42.3952 - val_accuracy: 0.7778
Epoch 31/40
22/22 [==============================] - 7s 296ms/step - loss: 41.3130 - accurac
y: 0.8333 - val_loss: 40.3870 - val_accuracy: 0.7778
Epoch 32/40
22/22 [==============================] - 7s 296ms/step - loss: 39.2677 - accurac
y: 0.8935 - val_loss: 38.4053 - val_accuracy: 0.7593
Epoch 33/40
22/22 [==============================] - 6s 291ms/step - loss: 37.3869 - accurac
```

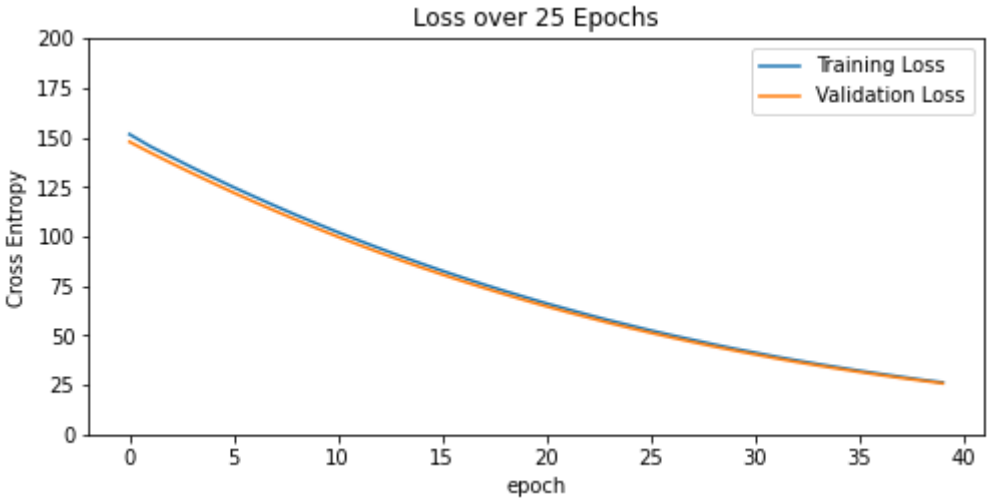
```
y: 0.8565 - val_loss: 36.5575 - val_accuracy: 0.7778
Epoch 34/40
22/22 [==============================] - 6s 279ms/step - loss: 35.5620 - accurac
y: 0.8704 - val_loss: 34.8284 - val_accuracy: 0.7593
Epoch 35/40
22/22 [==============================] - 6s 291ms/step - loss: 33.8723 - accurac
y: 0.8565 - val_loss: 33.1516 - val_accuracy: 0.7593
Epoch 36/40
22/22 [==============================] - 6s 281ms/step - loss: 32.1750 - accurac
y: 0.8519 - val_loss: 31.5512 - val_accuracy: 0.7778
Epoch 37/40
22/22 [==============================] - 6s 283ms/step - loss: 30.6147 - accurac
y: 0.8472 - val_loss: 29.9853 - val_accuracy: 0.7778
Epoch 38/40
22/22 [==============================] - 6s 286ms/step - loss: 29.0984 - accurac
y: 0.8796 - val_loss: 28.5072 - val_accuracy: 0.7778
Epoch 39/40
22/22 [==============================] - 6s 285ms/step - loss: 27.6067 - accurac
y: 0.9352 - val_loss: 27.1510 - val_accuracy: 0.7778
Epoch 40/40
22/22 [==============================] - 6s 287ms/step - loss: 26.2654 - accurac
y: 0.9167 - val_loss: 25.7764 - val_accuracy: 0.7778
```

[5 points] Plot Accuracy and Loss During Training

In [56]:

```
import matplotlib.pyplot as plt

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=' Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([.2,1.01])
plt.title('Accuracy over 25 Epochs')

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([0,200])
plt.title('Loss over 25 Epochs')
plt.xlabel('epoch')
plt.show()
```





Testing Model

In [57]:

```
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZ
                                                  batch_size=1,shuffle=True,seed
eval_generator.reset()
print(len(eval_generator))
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])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 1s 34ms/step - loss: 25.7542 - accurac
y: 0.7500
Test loss: 25.75415802001953
Test accuracy: 0.75
```

[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.

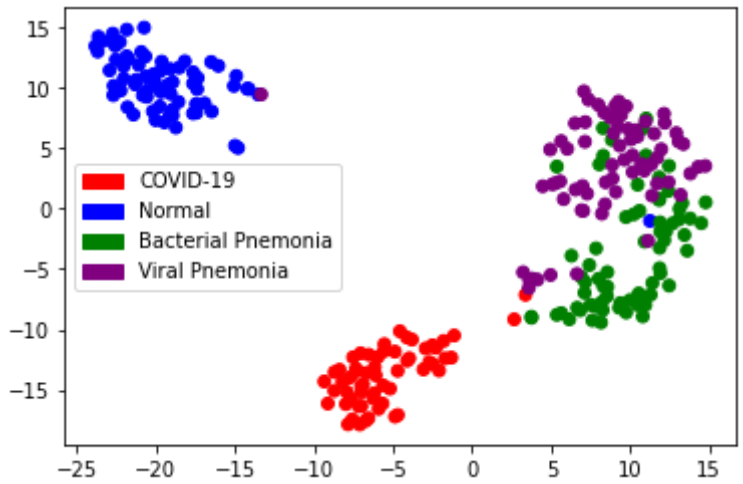
In [58]:
```
from tensorflow.keras import models
from sklearn.manifold import TSNE
import matplotlib.patches as mpatches
```

In [63]:
```
intermediate_layer_model = models.Model(inputs=model.input,
                                        outputs=model.get_layer('dense_16').outp

tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=
                                                       batch_size=1,shuffle=False,see

intermediate_output = intermediate_layer_model.predict(tsne_data_generator)

reducedData = TSNE(n_components=2).fit_transform(intermediate_output)

class_array = tsne_data_generator.labels

tsne_data_generator.class_indices
```

```
Found 270 images belonging to 4 classes.
```

Out[63]:
```
{'covid': 0, 'normal': 1, 'pneumonia_bac': 2, 'pneumonia_vir': 3}
```

In [64]:
```
colormapping = {0: 'red', 1: 'blue', 2: 'green', 3: 'purple'}
colors = [colormapping[i] for i in class_array]

plt.scatter(x = reducedData[:, 0], y = reducedData[:, 1], c = colors)

red_patch = mpatches.Patch(color='red', label='COVID-19')
blue_patch = mpatches.Patch(color='blue', label='Normal')
green_patch = mpatches.Patch(color='green', label='Bacterial Pnemonia')
purple_patch = mpatches.Patch(color='purple', label='Viral Pnemonia')
plt.legend(handles=[red_patch, blue_patch, green_patch, purple_patch])
plt.show()
```



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

Task 2 (Model 2) [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] Multi-class Classification

In [29]:
```
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[29]: '2.3.1'

Load Image Data

In [30]:
```
DATA_LIST = os.listdir('all/train')
DATASET_PATH  = 'all/train'
TEST_DIR =  'all/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 10  # try reducing batch size or freeze more layers if your GPU
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experiment wit
```

Generate Training and Validation Batches

In [31]:
```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_
                                   featurewise_std_normalization = True,width_sh
                                   height_shift_range=0.2,shear_range=0.25,zoom_
                                   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
                                                  shuffle=True,batch_size=BATCH_
                                                  subset = "training",seed=42,
                                                  class_mode="categorical")

valid_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE
                                                  shuffle=True,batch_size=BATCH_
                                                  subset = "validation",
                                                  seed=42,class_mode="categorica
```

```
/share/pkg.7/tensorflow/2.3.1/install/lib/SCC/../python3.6/site-packages/keras_p
reprocessing/image/image_data_generator.py:342: UserWarning: This ImageDataGener
ator specifies `zca_whitening` which overrides setting of`featurewise_std_normal
ization`.
  warnings.warn('This ImageDataGenerator specifies '
Found 216 images belonging to 4 classes.
Found 54 images belonging to 4 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 [35]:
```
IMG_SHAPE = IMAGE_SIZE + (3,)
base_model = tf.keras.applications.resnet50.ResNet50(input_shape=IMG_SHAPE,
                                                     include_top=False,
                                                     weights='imagenet')
base_model.trainable = True

print("Number of layers in the base model: ", len(base_model.layers))
```

```
# Fine-tune from this layer onwards
fine_tune_at = 172

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable =  False

base_model.summary()
```

```
Number of layers in the base model:  175
Model: "resnet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_14 (InputLayer)           [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_14[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0]
[0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0]
[0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool[0]
[0]
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_1_
conv[0][0]
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1_1_
bn[0][0]
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block1_1_
relu[0][0]
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_2_
conv[0][0]
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1_2_
bn[0][0]
__________________________________________________________________________________________________
```

```
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool[0]
[0]
__________________________________________________________________________________
__________________
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block1_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_0_
conv[0][0]
__________________________________________________________________________________
__________________
conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv2_block1_add (Add)          (None, 56, 56, 256)  0           conv2_block1_0_
bn[0][0]
                                                                 conv2_block1_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv2_block1_out (Activation)   (None, 56, 56, 256)  0           conv2_block1_ad
d[0][0]
__________________________________________________________________________________
__________________
conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block1_ou
t[0][0]
__________________________________________________________________________________
__________________
conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0           conv2_block2_1_
bn[0][0]
__________________________________________________________________________________
__________________
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block2_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0           conv2_block2_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block2_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block2_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv2_block2_add (Add)          (None, 56, 56, 256)  0           conv2_block1_ou
t[0][0]
                                                                 conv2_block2_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv2_block2_out (Activation)   (None, 56, 56, 256)  0           conv2_block2_ad
```

```
d[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block2_ou
t[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block3_1_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_1_relu (Activation (None, 56, 56, 64)   0           conv2_block3_1_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block3_1_
relu[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block3_2_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_2_relu (Activation (None, 56, 56, 64)   0           conv2_block3_2_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block3_2_
relu[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block3_3_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_add (Add)          (None, 56, 56, 256)  0           conv2_block2_ou
t[0][0]
                                                                 conv2_block3_3_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv2_block3_out (Activation)   (None, 56, 56, 256)  0           conv2_block3_ad
d[0][0]
__________________________________________________________________________________________________
__________________
conv3_block1_1_conv (Conv2D)    (None, 28, 28, 128)  32896       conv2_block3_ou
t[0][0]
__________________________________________________________________________________________________
__________________
conv3_block1_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block1_1_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block1_1_relu (Activation (None, 28, 28, 128)  0           conv3_block1_1_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block1_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block1_1_
relu[0][0]
__________________________________________________________________________________________________
__________________
conv3_block1_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block1_2_
conv[0][0]
__________________________________________________________________________________________________
__________________
```

```
conv3_block1_2_relu (Activation (None, 28, 28, 128)  0           conv3_block1_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv3_block1_0_conv (Conv2D)    (None, 28, 28, 512)  131584      conv2_block3_ou
t[0][0]
__________________________________________________________________________________
__________________
conv3_block1_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block1_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv3_block1_0_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block1_0_
conv[0][0]
__________________________________________________________________________________
__________________
conv3_block1_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block1_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv3_block1_add (Add)          (None, 28, 28, 512)  0           conv3_block1_0_
bn[0][0]
                                                                 conv3_block1_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv3_block1_out (Activation)   (None, 28, 28, 512)  0           conv3_block1_ad
d[0][0]
__________________________________________________________________________________
__________________
conv3_block2_1_conv (Conv2D)    (None, 28, 28, 128)  65664       conv3_block1_ou
t[0][0]
__________________________________________________________________________________
__________________
conv3_block2_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block2_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv3_block2_1_relu (Activation (None, 28, 28, 128)  0           conv3_block2_1_
bn[0][0]
__________________________________________________________________________________
__________________
conv3_block2_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block2_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv3_block2_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block2_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv3_block2_2_relu (Activation (None, 28, 28, 128)  0           conv3_block2_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv3_block2_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block2_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv3_block2_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block2_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv3_block2_add (Add)          (None, 28, 28, 512)  0           conv3_block1_ou
t[0][0]
                                                                 conv3_block2_3_
```

```
bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_out (Activation)   (None, 28, 28, 512)  0           conv3_block2_ad
d[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_1_conv (Conv2D)    (None, 28, 28, 128)  65664       conv3_block2_ou
t[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block3_1_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_1_relu (Activation (None, 28, 28, 128)  0           conv3_block3_1_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block3_1_
relu[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block3_2_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_2_relu (Activation (None, 28, 28, 128)  0           conv3_block3_2_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block3_2_
relu[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block3_3_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_add (Add)          (None, 28, 28, 512)  0           conv3_block2_ou
t[0][0]
                                                                 conv3_block3_3_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_out (Activation)   (None, 28, 28, 512)  0           conv3_block3_ad
d[0][0]
__________________________________________________________________________________________________
__________________
conv3_block4_1_conv (Conv2D)    (None, 28, 28, 128)  65664       conv3_block3_ou
t[0][0]
__________________________________________________________________________________________________
__________________
conv3_block4_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block4_1_
conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block4_1_relu (Activation (None, 28, 28, 128)  0           conv3_block4_1_
bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block4_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block4_1_
relu[0][0]
__________________________________________________________________________________________________
__________________
```

```
conv3_block4_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block4_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv3_block4_2_relu (Activation (None, 28, 28, 128)  0           conv3_block4_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv3_block4_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block4_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv3_block4_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block4_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv3_block4_add (Add)          (None, 28, 28, 512)  0           conv3_block3_ou
t[0][0]
                                                                 conv3_block4_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv3_block4_out (Activation)   (None, 28, 28, 512)  0           conv3_block4_ad
d[0][0]
__________________________________________________________________________________
__________________
conv4_block1_1_conv (Conv2D)    (None, 14, 14, 256)  131328      conv3_block4_ou
t[0][0]
__________________________________________________________________________________
__________________
conv4_block1_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block1_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block1_1_relu (Activation (None, 14, 14, 256)  0           conv4_block1_1_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block1_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block1_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block1_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block1_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block1_2_relu (Activation (None, 14, 14, 256)  0           conv4_block1_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block1_0_conv (Conv2D)    (None, 14, 14, 1024) 525312      conv3_block4_ou
t[0][0]
__________________________________________________________________________________
__________________
conv4_block1_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block1_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block1_0_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block1_0_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block1_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block1_3_
conv[0][0]
__________________________________________________________________________________
__________________
```

```
__________________
conv4_block1_add (Add)          (None, 14, 14, 1024) 0           conv4_block1_0_
bn[0][0]
                                                                 conv4_block1_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block1_out (Activation)   (None, 14, 14, 1024) 0           conv4_block1_ad
d[0][0]
__________________________________________________________________________________
__________________
conv4_block2_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block1_ou
t[0][0]
__________________________________________________________________________________
__________________
conv4_block2_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block2_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block2_1_relu (Activation (None, 14, 14, 256)  0           conv4_block2_1_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block2_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block2_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block2_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block2_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block2_2_relu (Activation (None, 14, 14, 256)  0           conv4_block2_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block2_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block2_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block2_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block2_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block2_add (Add)          (None, 14, 14, 1024) 0           conv4_block1_ou
t[0][0]
                                                                 conv4_block2_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block2_out (Activation)   (None, 14, 14, 1024) 0           conv4_block2_ad
d[0][0]
__________________________________________________________________________________
__________________
conv4_block3_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block2_ou
t[0][0]
__________________________________________________________________________________
__________________
conv4_block3_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block3_1_relu (Activation (None, 14, 14, 256)  0           conv4_block3_1_
bn[0][0]
__________________________________________________________________________________
__________________
```

```
conv4_block3_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block3_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block3_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block3_2_relu (Activation (None, 14, 14, 256)  0           conv4_block3_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block3_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block3_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block3_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block3_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block3_add (Add)          (None, 14, 14, 1024) 0           conv4_block2_ou
t[0][0]
                                                                 conv4_block3_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block3_out (Activation)   (None, 14, 14, 1024) 0           conv4_block3_ad
d[0][0]
__________________________________________________________________________________
__________________
conv4_block4_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block3_ou
t[0][0]
__________________________________________________________________________________
__________________
conv4_block4_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block4_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block4_1_relu (Activation (None, 14, 14, 256)  0           conv4_block4_1_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block4_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block4_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block4_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block4_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block4_2_relu (Activation (None, 14, 14, 256)  0           conv4_block4_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block4_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block4_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block4_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block4_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block4_add (Add)          (None, 14, 14, 1024) 0           conv4_block3_ou
t[0][0]
                                                                 conv4_block4_3_
```

```
bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block4_out (Activation)   (None, 14, 14, 1024) 0           conv4_block4_ad
d[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block4_ou
t[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block5_1_
conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_1_relu (Activation (None, 14, 14, 256)  0           conv4_block5_1_
bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block5_1_
relu[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block5_2_
conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_2_relu (Activation (None, 14, 14, 256)  0           conv4_block5_2_
bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block5_2_
relu[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block5_3_
conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_add (Add)          (None, 14, 14, 1024) 0           conv4_block4_ou
t[0][0]
                                                                 conv4_block5_3_

bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_out (Activation)   (None, 14, 14, 1024) 0           conv4_block5_ad
d[0][0]
__________________________________________________________________________________________________
____________________
conv4_block6_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block5_ou
t[0][0]
__________________________________________________________________________________________________
____________________
conv4_block6_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block6_1_
conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block6_1_relu (Activation (None, 14, 14, 256)  0           conv4_block6_1_
bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block6_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block6_1_
relu[0][0]
__________________________________________________________________________________________________
____________________
```

```
conv4_block6_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block6_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block6_2_relu (Activation (None, 14, 14, 256)  0           conv4_block6_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block6_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block6_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv4_block6_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block6_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv4_block6_add (Add)          (None, 14, 14, 1024) 0           conv4_block5_ou
t[0][0]
                                                                 conv4_block6_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv4_block6_out (Activation)   (None, 14, 14, 1024) 0           conv4_block6_ad
d[0][0]
__________________________________________________________________________________
__________________
conv5_block1_1_conv (Conv2D)    (None, 7, 7, 512)    524800      conv4_block6_ou
t[0][0]
__________________________________________________________________________________
__________________
conv5_block1_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block1_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block1_1_relu (Activation (None, 7, 7, 512)    0           conv5_block1_1_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block1_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block1_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv5_block1_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block1_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block1_2_relu (Activation (None, 7, 7, 512)    0           conv5_block1_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block1_0_conv (Conv2D)    (None, 7, 7, 2048)   2099200     conv4_block6_ou
t[0][0]
__________________________________________________________________________________
__________________
conv5_block1_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block1_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv5_block1_0_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block1_0_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block1_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block1_3_
conv[0][0]
__________________________________________________________________________________
```

```
__________________
conv5_block1_add (Add)          (None, 7, 7, 2048)   0           conv5_block1_0_
bn[0][0]
                                                                 conv5_block1_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block1_out (Activation)   (None, 7, 7, 2048)   0           conv5_block1_ad
d[0][0]
__________________________________________________________________________________
__________________
conv5_block2_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_block1_ou
t[0][0]
__________________________________________________________________________________
__________________
conv5_block2_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block2_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block2_1_relu (Activation (None, 7, 7, 512)    0           conv5_block2_1_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block2_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block2_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv5_block2_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block2_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block2_2_relu (Activation (None, 7, 7, 512)    0           conv5_block2_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block2_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block2_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv5_block2_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block2_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block2_add (Add)          (None, 7, 7, 2048)   0           conv5_block1_ou
t[0][0]
                                                                 conv5_block2_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block2_out (Activation)   (None, 7, 7, 2048)   0           conv5_block2_ad
d[0][0]
__________________________________________________________________________________
__________________
conv5_block3_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_block2_ou
t[0][0]
__________________________________________________________________________________
__________________
conv5_block3_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3_1_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block3_1_relu (Activation (None, 7, 7, 512)    0           conv5_block3_1_
bn[0][0]
__________________________________________________________________________________
__________________
```

```
conv5_block3_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block3_1_
relu[0][0]
__________________________________________________________________________________
__________________
conv5_block3_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3_2_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block3_2_relu (Activation (None, 7, 7, 512)    0           conv5_block3_2_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block3_2_
relu[0][0]
__________________________________________________________________________________
__________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3_3_
conv[0][0]
__________________________________________________________________________________
__________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2_ou
t[0][0]
                                                                 conv5_block3_3_
bn[0][0]
__________________________________________________________________________________
__________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3_ad
d[0][0]
==================================================================================
==================
Total params: 23,587,712
Trainable params: 4,096
Non-trainable params: 23,583,616
__________________________________________________________________________________
__________________
```

In [37]:

```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = base_model(inputs, training=False)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(256, activation = 'relu')(x)
outputs = tf.keras.layers.Dense(4, activation='softmax')(x)
model = tf.keras.Model(inputs, outputs)

model.summary()
```

```
Model: "functional_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_16 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten_7 (Flatten)          (None, 100352)            0
_________________________________________________________________
dense_14 (Dense)             (None, 256)               25690368
_________________________________________________________________
dense_15 (Dense)             (None, 4)                 1028
=================================================================
Total params: 49,279,108
Trainable params: 25,695,492
Non-trainable params: 23,583,616
_________________________________________________________________
```

[5 points] Train Model

In [42]:

```
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(from_logits=False),
              optimizer = tf.keras.optimizers.RMSprop(lr=LEARNING_RATE),
              metrics=['accuracy'])

history = model.fit(train_batches,
                    epochs=100,
                    validation_data=valid_batches)
```

```
Epoch 1/100
22/22 [==============================] - 7s 312ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 2/100
22/22 [==============================] - 6s 272ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 3/100
22/22 [==============================] - 6s 287ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 4/100
22/22 [==============================] - 6s 285ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 5/100
22/22 [==============================] - 6s 285ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 6/100
22/22 [==============================] - 6s 293ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 7/100
22/22 [==============================] - 7s 295ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 8/100
22/22 [==============================] - 7s 303ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 9/100
22/22 [==============================] - 7s 301ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 10/100
22/22 [==============================] - 6s 295ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 11/100
22/22 [==============================] - 6s 290ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 12/100
22/22 [==============================] - 6s 288ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 13/100
22/22 [==============================] - 6s 291ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 14/100
22/22 [==============================] - 6s 285ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 15/100
22/22 [==============================] - 6s 293ms/step - loss: 1.3849 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 16/100
22/22 [==============================] - 6s 290ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 17/100
22/22 [==============================] - 7s 301ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 18/100
```

```
22/22 [==============================] - 6s 293ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 19/100
22/22 [==============================] - 6s 291ms/step - loss: 1.3899 - accurac
y: 0.2778 - val_loss: 1.3857 - val_accuracy: 0.2593
Epoch 20/100
22/22 [==============================] - 6s 289ms/step - loss: 1.3885 - accurac
y: 0.2917 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 21/100
22/22 [==============================] - 7s 296ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3844 - val_accuracy: 0.2593
Epoch 22/100
22/22 [==============================] - 6s 295ms/step - loss: 1.3835 - accurac
y: 0.2778 - val_loss: 1.3914 - val_accuracy: 0.2593
Epoch 23/100
22/22 [==============================] - 6s 293ms/step - loss: 1.3869 - accurac
y: 0.2454 - val_loss: 1.3843 - val_accuracy: 0.2593
Epoch 24/100
22/22 [==============================] - 6s 294ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3858 - val_accuracy: 0.2407
Epoch 25/100
22/22 [==============================] - 6s 287ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3834 - val_accuracy: 0.2778
Epoch 26/100
22/22 [==============================] - 6s 294ms/step - loss: 1.3891 - accurac
y: 0.2685 - val_loss: 1.3807 - val_accuracy: 0.2593
Epoch 27/100
22/22 [==============================] - 6s 293ms/step - loss: 1.3846 - accurac
y: 0.2500 - val_loss: 1.3843 - val_accuracy: 0.2593
Epoch 28/100
22/22 [==============================] - 6s 291ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3843 - val_accuracy: 0.2593
Epoch 29/100
22/22 [==============================] - 6s 279ms/step - loss: 1.3844 - accurac
y: 0.2593 - val_loss: 1.3843 - val_accuracy: 0.2593
Epoch 30/100
22/22 [==============================] - 6s 289ms/step - loss: 1.3883 - accurac
y: 0.2546 - val_loss: 1.3847 - val_accuracy: 0.2593
Epoch 31/100
22/22 [==============================] - 6s 286ms/step - loss: 1.3849 - accurac
y: 0.2639 - val_loss: 1.3843 - val_accuracy: 0.2593
Epoch 32/100
22/22 [==============================] - 7s 301ms/step - loss: 1.3908 - accurac
y: 0.2546 - val_loss: 1.3843 - val_accuracy: 0.2593
Epoch 33/100
22/22 [==============================] - 6s 289ms/step - loss: 1.3845 - accurac
y: 0.2593 - val_loss: 1.3843 - val_accuracy: 0.2593
Epoch 34/100
22/22 [==============================] - 6s 294ms/step - loss: 1.3929 - accurac
y: 0.2731 - val_loss: 1.3829 - val_accuracy: 0.2778
Epoch 35/100
22/22 [==============================] - 7s 301ms/step - loss: 1.3467 - accurac
y: 0.2778 - val_loss: 1.3699 - val_accuracy: 0.2593
Epoch 36/100
22/22 [==============================] - 6s 286ms/step - loss: 1.3395 - accurac
y: 0.2778 - val_loss: 1.3630 - val_accuracy: 0.3333
Epoch 37/100
22/22 [==============================] - 6s 295ms/step - loss: 1.3376 - accurac
y: 0.3750 - val_loss: 1.2926 - val_accuracy: 0.4630
Epoch 38/100
22/22 [==============================] - 6s 295ms/step - loss: 1.2997 - accurac
y: 0.4491 - val_loss: 1.3135 - val_accuracy: 0.3704
Epoch 39/100
22/22 [==============================] - 7s 298ms/step - loss: 1.3198 - accurac
y: 0.4120 - val_loss: 1.2839 - val_accuracy: 0.4259
```

```
Epoch 40/100
22/22 [==============================] - 6s 288ms/step - loss: 1.2994 - accurac
y: 0.4583 - val_loss: 1.3524 - val_accuracy: 0.4444
Epoch 41/100
22/22 [==============================] - 7s 308ms/step - loss: 1.2770 - accurac
y: 0.4630 - val_loss: 1.2987 - val_accuracy: 0.4074
Epoch 42/100
22/22 [==============================] - 6s 290ms/step - loss: 1.2880 - accurac
y: 0.4306 - val_loss: 1.3382 - val_accuracy: 0.3519
Epoch 43/100
22/22 [==============================] - 6s 282ms/step - loss: 1.2843 - accurac
y: 0.4583 - val_loss: 1.2630 - val_accuracy: 0.4630
Epoch 44/100
22/22 [==============================] - 6s 284ms/step - loss: 1.2655 - accurac
y: 0.4954 - val_loss: 1.3038 - val_accuracy: 0.4630
Epoch 45/100
22/22 [==============================] - 6s 289ms/step - loss: 1.2874 - accurac
y: 0.4306 - val_loss: 1.3122 - val_accuracy: 0.4630
Epoch 46/100
22/22 [==============================] - 7s 300ms/step - loss: 1.2641 - accurac
y: 0.4352 - val_loss: 1.2440 - val_accuracy: 0.5000
Epoch 47/100
22/22 [==============================] - 7s 297ms/step - loss: 1.2740 - accurac
y: 0.4676 - val_loss: 1.3133 - val_accuracy: 0.3889
Epoch 48/100
22/22 [==============================] - 6s 275ms/step - loss: 1.2688 - accurac
y: 0.4491 - val_loss: 1.2530 - val_accuracy: 0.5556
Epoch 49/100
22/22 [==============================] - 6s 291ms/step - loss: 1.2398 - accurac
y: 0.4907 - val_loss: 1.2985 - val_accuracy: 0.3704
Epoch 50/100
22/22 [==============================] - 6s 285ms/step - loss: 1.2514 - accurac
y: 0.4398 - val_loss: 1.2528 - val_accuracy: 0.4815
Epoch 51/100
22/22 [==============================] - 6s 283ms/step - loss: 1.2599 - accurac
y: 0.4167 - val_loss: 1.2578 - val_accuracy: 0.4074
Epoch 52/100
22/22 [==============================] - 6s 289ms/step - loss: 1.2490 - accurac
y: 0.5000 - val_loss: 1.2614 - val_accuracy: 0.4259
Epoch 53/100
22/22 [==============================] - 6s 287ms/step - loss: 1.2385 - accurac
y: 0.4583 - val_loss: 1.3067 - val_accuracy: 0.3704
Epoch 54/100
22/22 [==============================] - 6s 283ms/step - loss: 1.2335 - accurac
y: 0.4630 - val_loss: 1.2613 - val_accuracy: 0.4074
Epoch 55/100
22/22 [==============================] - 6s 284ms/step - loss: 1.2137 - accurac
y: 0.4676 - val_loss: 1.2614 - val_accuracy: 0.4444
Epoch 56/100
22/22 [==============================] - 6s 289ms/step - loss: 1.2479 - accurac
y: 0.4074 - val_loss: 1.2594 - val_accuracy: 0.4074
Epoch 57/100
22/22 [==============================] - 6s 283ms/step - loss: 1.2304 - accurac
y: 0.4583 - val_loss: 1.2254 - val_accuracy: 0.4444
Epoch 58/100
22/22 [==============================] - 7s 300ms/step - loss: 1.2405 - accurac
y: 0.4583 - val_loss: 1.2021 - val_accuracy: 0.5556
Epoch 59/100
22/22 [==============================] - 6s 294ms/step - loss: 1.1883 - accurac
y: 0.4676 - val_loss: 1.2069 - val_accuracy: 0.5185
Epoch 60/100
22/22 [==============================] - 6s 288ms/step - loss: 1.2250 - accurac
y: 0.4583 - val_loss: 1.2257 - val_accuracy: 0.3889
Epoch 61/100
22/22 [==============================] - 7s 298ms/step - loss: 1.2192 - accurac
```

```
y: 0.4537 - val_loss: 1.1670 - val_accuracy: 0.5000
Epoch 62/100
22/22 [==============================] - 6s 289ms/step - loss: 1.2021 - accurac
y: 0.4676 - val_loss: 1.2112 - val_accuracy: 0.4444
Epoch 63/100
22/22 [==============================] - 6s 277ms/step - loss: 1.2276 - accurac
y: 0.4676 - val_loss: 1.2255 - val_accuracy: 0.4444
Epoch 64/100
22/22 [==============================] - 6s 283ms/step - loss: 1.2119 - accurac
y: 0.4259 - val_loss: 1.2089 - val_accuracy: 0.3333
Epoch 65/100
22/22 [==============================] - 6s 289ms/step - loss: 1.2164 - accurac
y: 0.4722 - val_loss: 1.2701 - val_accuracy: 0.3148
Epoch 66/100
22/22 [==============================] - 6s 288ms/step - loss: 1.2034 - accurac
y: 0.4769 - val_loss: 1.1801 - val_accuracy: 0.4444
Epoch 67/100
22/22 [==============================] - 6s 284ms/step - loss: 1.1839 - accurac
y: 0.4861 - val_loss: 1.2190 - val_accuracy: 0.4630
Epoch 68/100
22/22 [==============================] - 6s 289ms/step - loss: 1.2098 - accurac
y: 0.4722 - val_loss: 1.1780 - val_accuracy: 0.5185
Epoch 69/100
22/22 [==============================] - 6s 286ms/step - loss: 1.1906 - accurac
y: 0.4630 - val_loss: 1.2121 - val_accuracy: 0.3889
Epoch 70/100
22/22 [==============================] - 6s 283ms/step - loss: 1.2107 - accurac
y: 0.4630 - val_loss: 1.1490 - val_accuracy: 0.5185
Epoch 71/100
22/22 [==============================] - 6s 289ms/step - loss: 1.2045 - accurac
y: 0.4815 - val_loss: 1.2440 - val_accuracy: 0.4074
Epoch 72/100
22/22 [==============================] - 6s 282ms/step - loss: 1.1983 - accurac
y: 0.5046 - val_loss: 1.1688 - val_accuracy: 0.4444
Epoch 73/100
22/22 [==============================] - 6s 291ms/step - loss: 1.1708 - accurac
y: 0.5185 - val_loss: 1.1243 - val_accuracy: 0.5185
Epoch 74/100
22/22 [==============================] - 6s 288ms/step - loss: 1.1810 - accurac
y: 0.4769 - val_loss: 1.2002 - val_accuracy: 0.4444
Epoch 75/100
22/22 [==============================] - 6s 287ms/step - loss: 1.1795 - accurac
y: 0.4537 - val_loss: 1.1386 - val_accuracy: 0.5185
Epoch 76/100
22/22 [==============================] - 6s 290ms/step - loss: 1.2070 - accurac
y: 0.4537 - val_loss: 1.2236 - val_accuracy: 0.4259
Epoch 77/100
22/22 [==============================] - 6s 289ms/step - loss: 1.1781 - accurac
y: 0.5093 - val_loss: 1.1714 - val_accuracy: 0.4444
Epoch 78/100
22/22 [==============================] - 6s 287ms/step - loss: 1.1537 - accurac
y: 0.4954 - val_loss: 1.1349 - val_accuracy: 0.4444
Epoch 79/100
22/22 [==============================] - 6s 268ms/step - loss: 1.1663 - accurac
y: 0.4630 - val_loss: 1.1868 - val_accuracy: 0.4259
Epoch 80/100
22/22 [==============================] - 6s 291ms/step - loss: 1.1886 - accurac
y: 0.4676 - val_loss: 1.1355 - val_accuracy: 0.5000
Epoch 81/100
22/22 [==============================] - 6s 286ms/step - loss: 1.1765 - accurac
y: 0.4815 - val_loss: 1.1695 - val_accuracy: 0.3704
Epoch 82/100
22/22 [==============================] - 6s 285ms/step - loss: 1.1864 - accurac
y: 0.5139 - val_loss: 1.1857 - val_accuracy: 0.4444
Epoch 83/100
```

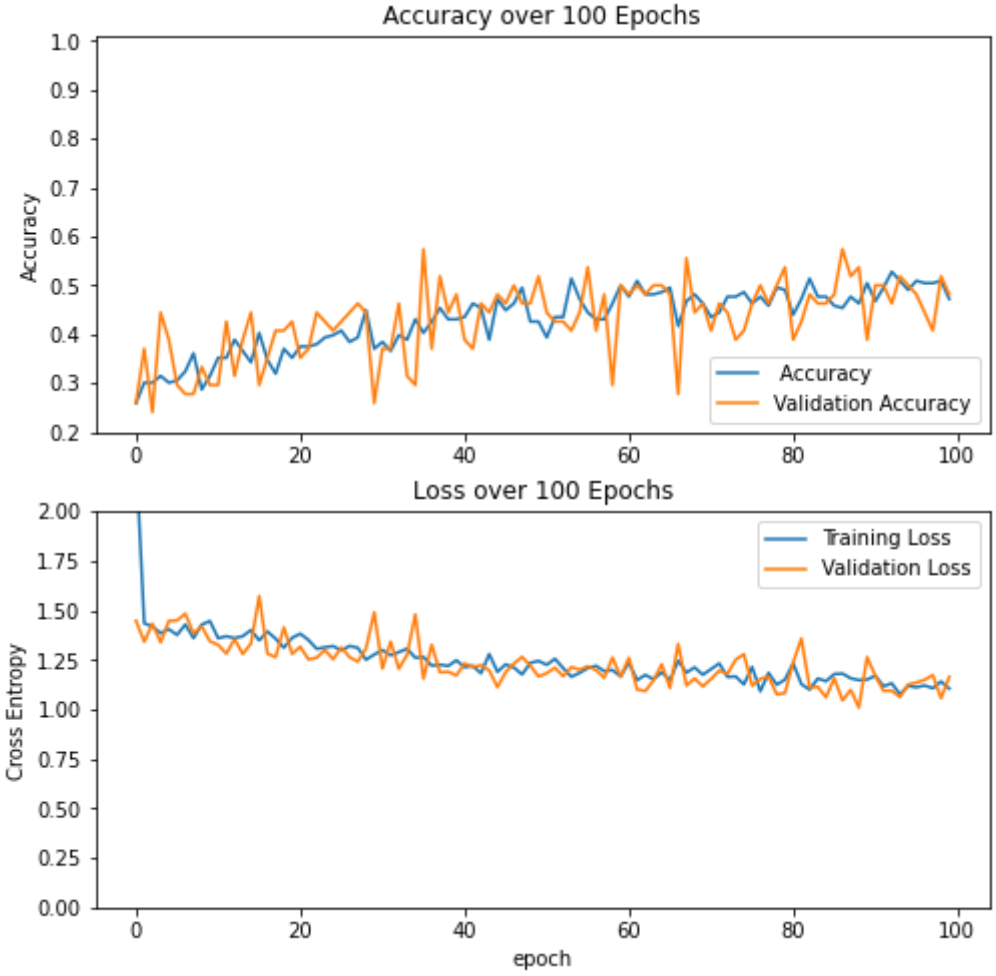
```
22/22 [==============================] - 6s 288ms/step - loss: 1.1738 - accurac
y: 0.4722 - val_loss: 1.1579 - val_accuracy: 0.5000
Epoch 84/100
22/22 [==============================] - 6s 290ms/step - loss: 1.1608 - accurac
y: 0.5000 - val_loss: 1.2581 - val_accuracy: 0.4444
Epoch 85/100
22/22 [==============================] - 6s 291ms/step - loss: 1.1754 - accurac
y: 0.4491 - val_loss: 1.1347 - val_accuracy: 0.4630
Epoch 86/100
22/22 [==============================] - 6s 289ms/step - loss: 1.1521 - accurac
y: 0.4630 - val_loss: 1.1298 - val_accuracy: 0.5000
Epoch 87/100
22/22 [==============================] - 6s 293ms/step - loss: 1.1690 - accurac
y: 0.4722 - val_loss: 1.1465 - val_accuracy: 0.4074
Epoch 88/100
22/22 [==============================] - 6s 295ms/step - loss: 1.1332 - accurac
y: 0.4954 - val_loss: 1.1267 - val_accuracy: 0.5556
Epoch 89/100
22/22 [==============================] - 6s 282ms/step - loss: 1.1297 - accurac
y: 0.5000 - val_loss: 1.1551 - val_accuracy: 0.4444
Epoch 90/100
22/22 [==============================] - 7s 298ms/step - loss: 1.1787 - accurac
y: 0.4583 - val_loss: 1.1541 - val_accuracy: 0.4630
Epoch 91/100
22/22 [==============================] - 6s 291ms/step - loss: 1.1228 - accurac
y: 0.5185 - val_loss: 1.1327 - val_accuracy: 0.5185
Epoch 92/100
22/22 [==============================] - 7s 298ms/step - loss: 1.1648 - accurac
y: 0.5185 - val_loss: 1.1022 - val_accuracy: 0.5556
Epoch 93/100
22/22 [==============================] - 7s 297ms/step - loss: 1.1308 - accurac
y: 0.4861 - val_loss: 1.1212 - val_accuracy: 0.5370
Epoch 94/100
22/22 [==============================] - 6s 290ms/step - loss: 1.1506 - accurac
y: 0.4583 - val_loss: 1.1493 - val_accuracy: 0.4815
Epoch 95/100
22/22 [==============================] - 6s 284ms/step - loss: 1.1496 - accurac
y: 0.4722 - val_loss: 1.1374 - val_accuracy: 0.5556
Epoch 96/100
22/22 [==============================] - 6s 290ms/step - loss: 1.1294 - accurac
y: 0.4861 - val_loss: 1.1697 - val_accuracy: 0.4444
Epoch 97/100
22/22 [==============================] - 6s 289ms/step - loss: 1.1346 - accurac
y: 0.5046 - val_loss: 1.1269 - val_accuracy: 0.5741
Epoch 98/100
22/22 [==============================] - 6s 293ms/step - loss: 1.1294 - accurac
y: 0.5139 - val_loss: 1.1429 - val_accuracy: 0.4259
Epoch 99/100
22/22 [==============================] - 7s 296ms/step - loss: 1.1595 - accurac
y: 0.5046 - val_loss: 1.1557 - val_accuracy: 0.5185
Epoch 100/100
22/22 [==============================] - 6s 291ms/step - loss: 1.1387 - accurac
y: 0.4769 - val_loss: 1.2359 - val_accuracy: 0.4259
```

[5 points] Plot Accuracy and Loss During Training

In [40]:

```
import matplotlib.pyplot as plt

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=' Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([.2,1.01])
plt.title('Accuracy over 100 Epochs')

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([0,2.0])
plt.title('Loss over 100 Epochs')
plt.xlabel('epoch')
plt.show()
```



Testing Model

In [43]:

```
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZ
                                                  batch_size=1,shuffle=True,seed
eval_generator.reset()
print(len(eval_generator))
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])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 2s 44ms/step - loss: 1.4140 - accuracy:
0.2778
Test loss: 1.4140089750289917
Test accuracy: 0.2777777910232544
```

[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.

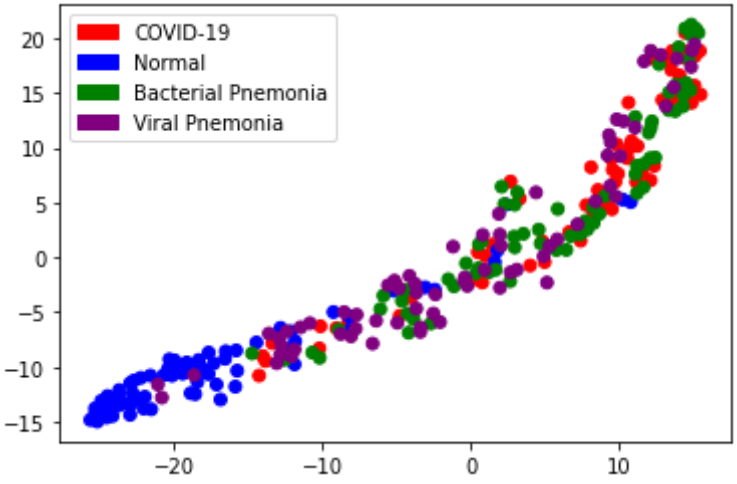
In [44]:
```
from tensorflow.keras import models
from sklearn.manifold import TSNE
import matplotlib.patches as mpatches
```

In [47]:
```
intermediate_layer_model = models.Model(inputs=model.input,
                                        outputs=model.get_layer('dense_14').outp

tsne_data_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=
                                                       batch_size=1,shuffle=False,see

intermediate_output = intermediate_layer_model.predict(tsne_data_generator)

reducedData = TSNE(n_components=2).fit_transform(intermediate_output)

class_array = tsne_data_generator.labels

tsne_data_generator.class_indices
```

```
Found 270 images belonging to 4 classes.
```

Out[47]:
```
{'covid': 0, 'normal': 1, 'pneumonia_bac': 2, 'pneumonia_vir': 3}
```

In [48]:
```
colormapping = {0: 'red', 1: 'blue', 2: 'green', 3: 'purple'}
colors = [colormapping[i] for i in class_array]

plt.scatter(x = reducedData[:, 0], y = reducedData[:, 1], c = colors)

red_patch = mpatches.Patch(color='red', label='COVID-19')
blue_patch = mpatches.Patch(color='blue', label='Normal')
green_patch = mpatches.Patch(color='green', label='Bacterial Pnemonia')
purple_patch = mpatches.Patch(color='purple', label='Viral Pnemonia')
plt.legend(handles=[red_patch, blue_patch, green_patch, purple_patch])
plt.show()
```


COVID-19
Normal
Bacterial Pnemonia
Viral Pnemonia
20
15
10
5
0
−5
−10
−15
−20
−10
0
10