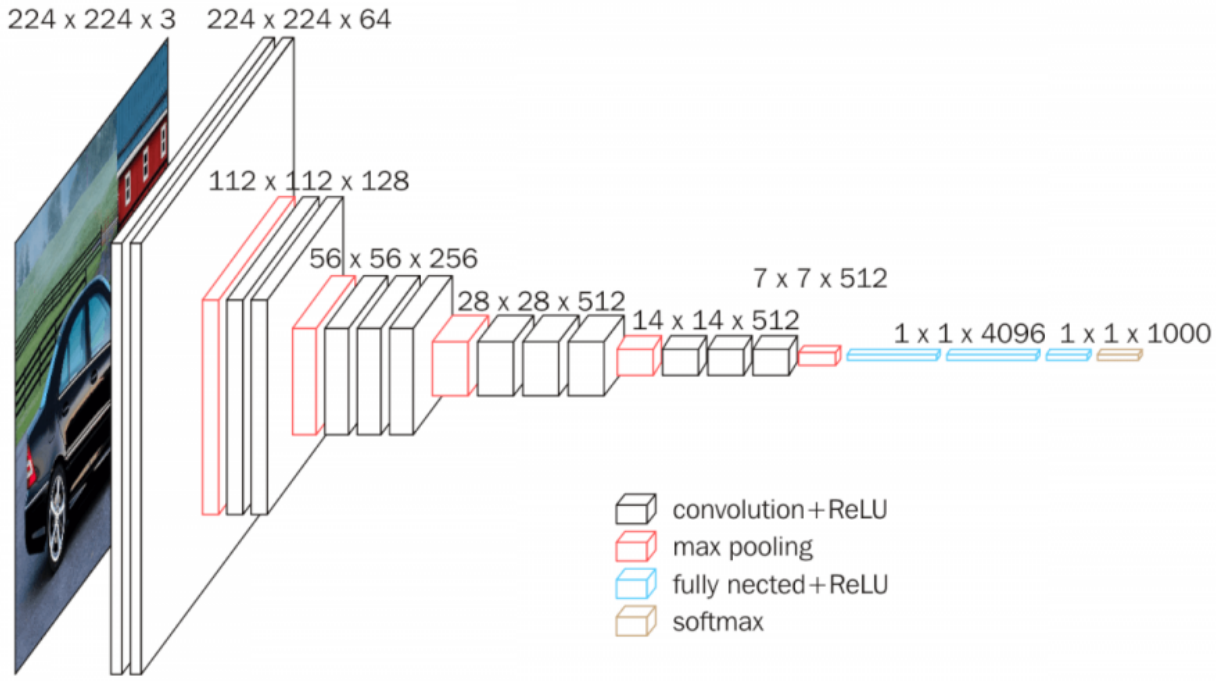
runsheng@bu.edu
jli0126@bu.edu

CS440 Class Challenge

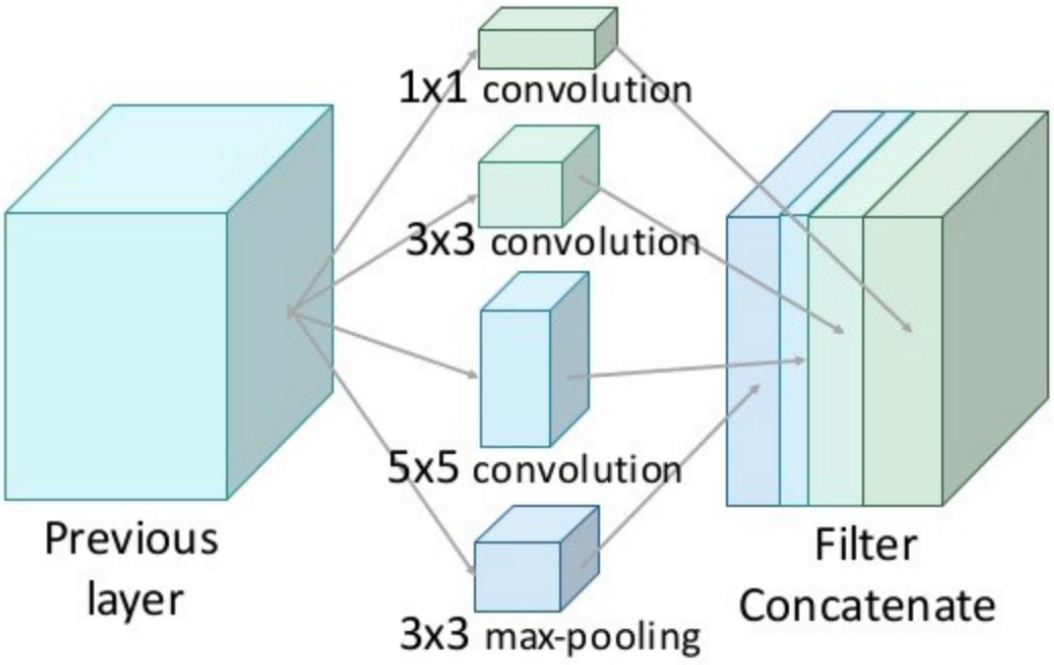
Architecture Description

Layers

- Task1 uses VGG16 as the pre-trained/base model. A Flatten layer immediately follows VGG16, then a fully connected layer with 256 dimensions were passed in and the ReLu activation function was usd, followed by another fully connected layer with one output dimension, with the sigmoid activation function being used. The VGG16 architecture is attached below.



- Task2 attempted to use two models, namely the VGG19 and the Xception. Both models were followed by a Flatten layer, a fully connected layer with 256 dimensions activated by the ReLu function, and another fully connected layer with four output dimensions activated by the softmax function. The Xception architecture is attached below.



Optimizer

- Both tasks uses the Adam optimizer, which combines properties of the AdaGrad and RMSProp algorithms handle sparse gradients

Loss Functions

- Task1 uses the binary cross-entropy loss function because it is a two-class problem
- Task 2 uses the categorical cross-entropy loss function because it is a multi-class problems (4 classes)

Parameters

- The parameters used in the pre-trained networks for both task1 and task2 are: (weights='imagenet', include_top=False, input_shape=(224, 224, 3). We are using the ImageNet weights to save training time (compared to random weights), and we are setting include_top = False to omit the final fully connected layer, as we will be incorporating this pre-trained model as part of our customized neural network. The input shape is 224 by 224 by 3, indicating the presence of RGB colors. The parameters in the subsequent three customized layers are the same for both tasks and for both pre-trained algorithms in task2.
- There are also a few other model parameters. The batch size is set to be 10 (using the default given in the starter code). The number of epochs is 40 in task1 and 100 in task2. The learning rates are 0.0005 and 0.0001 for task1 and task2, respectively

Regularization Techniques

- Within the preprocessing step (using the code included), data augmentation was applied. Specifically, rotation, horizontal flip, height shift, zoom, shear, and width shift. We experimented with dropout and fine-tuning the last three convolutional layers, both had bad results probably because the training dataset is too small, so we didn't use other techniques.

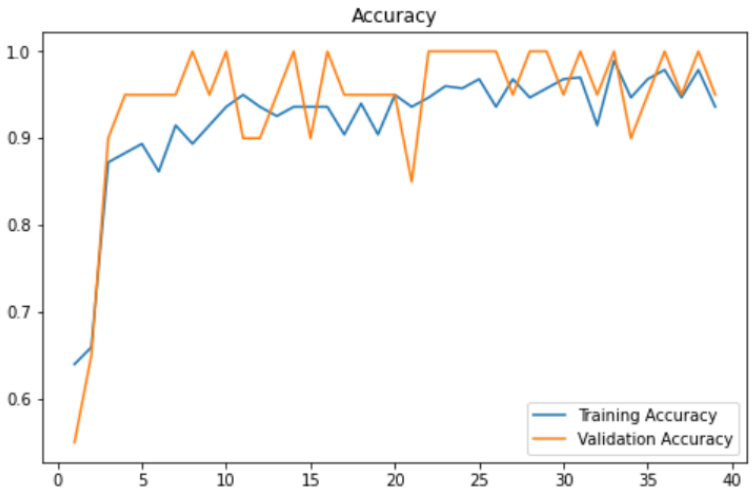
Performance Comparison

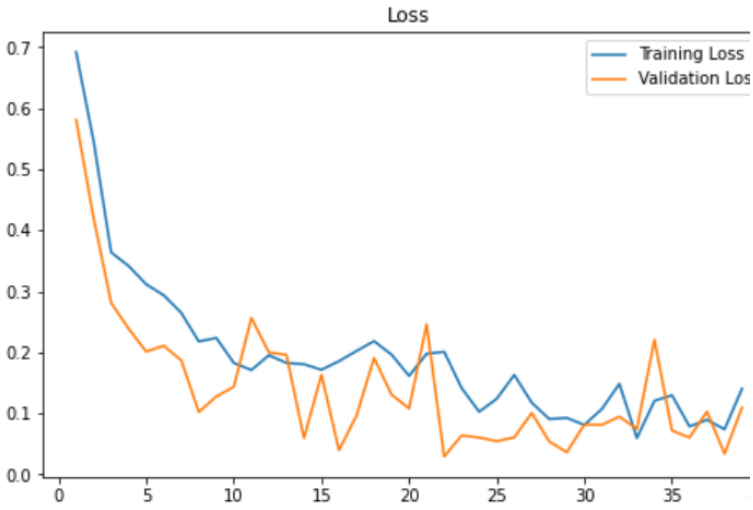
In task2, two architectures were experimented: VGG19 and Xception. Same parameter settings were used to be able to make a leveled comparison. We previously tried to set the last three convolutional layers of both the VGG19 and the Xception to be trainable, but both didn't give us good results. However, we noticed one difference while training weights on these pre-trained models. Xception is a lot quicker to train compared to VGG19. We think it is because convolutional filters in the Xception build up slower as compared to VGG19. Also, VGG19 computes convolution on the entire image on the first few layers, compared to Xception. Xception tends to go deepers and thinner compared to VGG19, which could explain its faster training time.

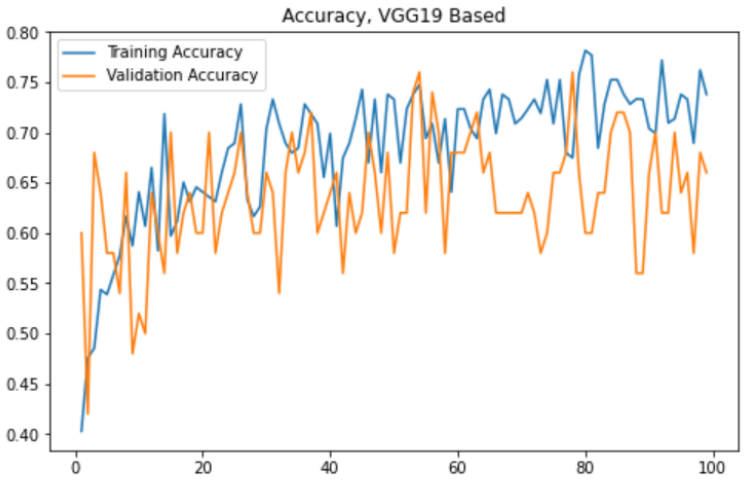
Upon researching VGG19 and Xception, we found that VGG19 could experience the problem of the "vanishing gradient", not as a result of sigmoid-like non-linearities, but as a result of the depth of the network. As backpropagation is performed, the gradients keep getting multiplied by each local gradient, resulting in the updates to the initial layers to be very small. This effect also increases the runtime of VGG19. From an accuracy/loss perspective, VGG19 performs worse than Xception. We would choose Xception for the final model where the pre-trained weights are fixed as compared to trainable weights.

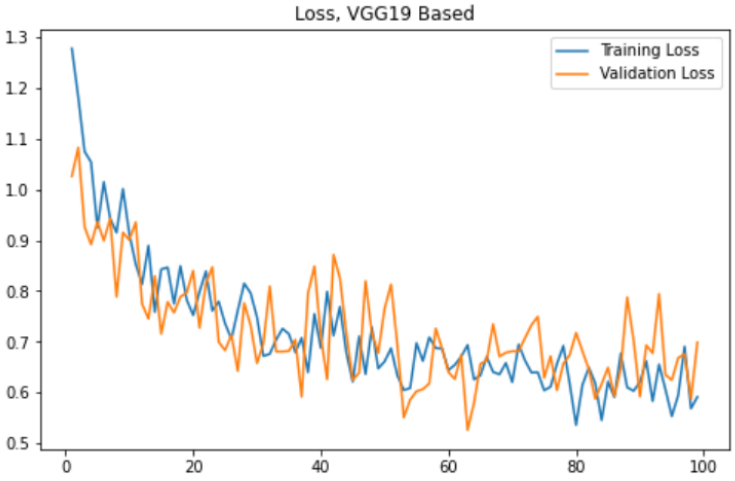
Accuracy and Loss

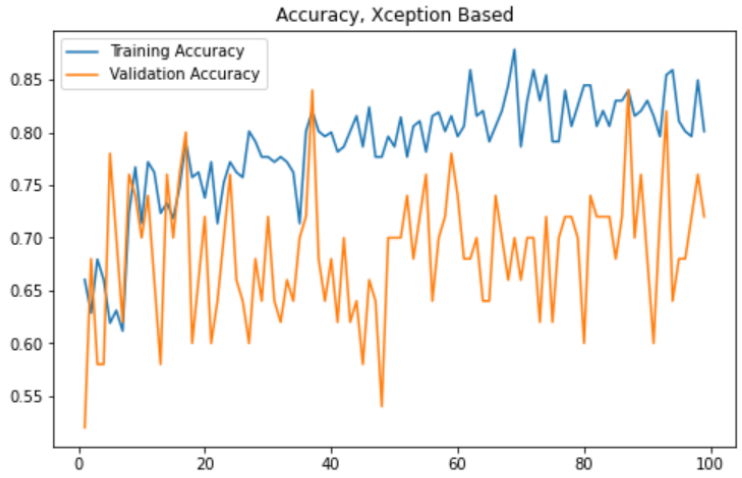
The accuracy and loss plots are attached here, which can also be found in the compiled notebook PDF files later.

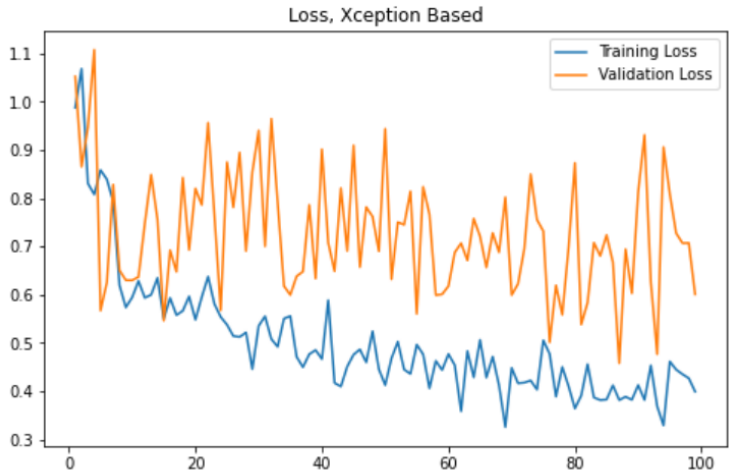



Loss
Training Loss
Validation Los
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
30
35


Accuracy, VGG19 Based
Training Accuracy
Validation Accuracy
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
20
40
60
80
100


Loss, VGG19 Based
Training Loss
Validation Loss
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100



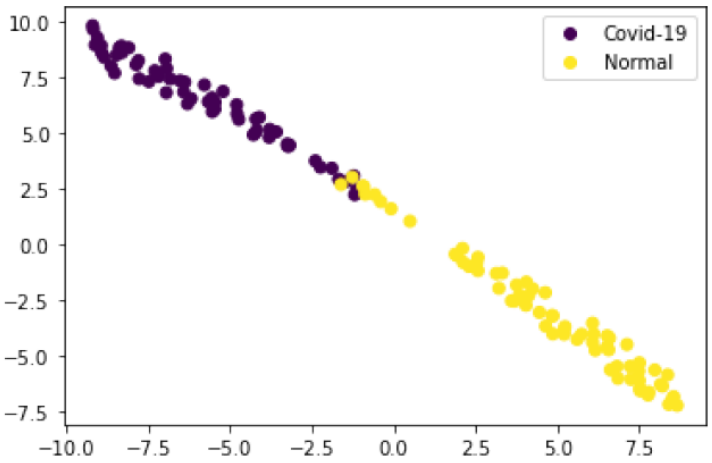


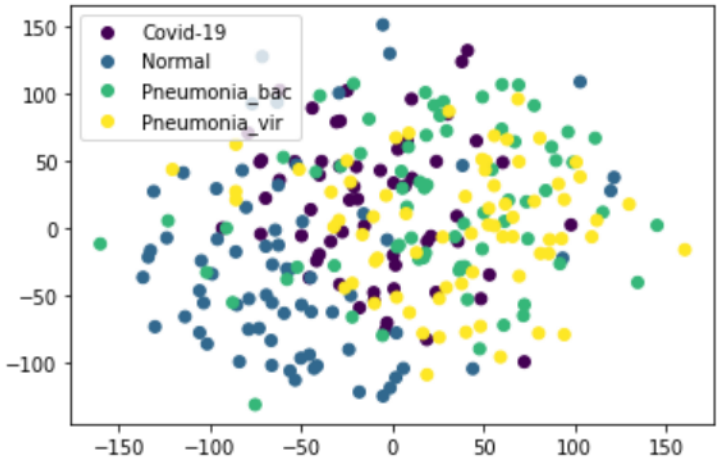
For task1, the accuracy increased sharply in the first five iterations and stabilized there in the subsequent iterations. The accuracy didn't drop, which means the model is still learning and performing well, and that the learning rate is appropriate. The loss seems to be steadily decreasing throughout all the epochs. There is evidence that validation loss stabilizes at around epoch = 35 and increases from epoch 35 to 40, indicating a potential overfit scenario for the model. However, there is no apparent sign of overfitting.

For task2, both models exhibit a similar phenomenon. There appears to be a gap between training accuracy and validation accuracy, with validation accuracy being quite lower than training accuracy. However, both accuracies are increasing, indicating the model is learning, but it is not learning very well. This problem might be fixed by changing the parameters of the pre-trained model, adding more layers into the network, or allowing pre-trained weights to be trained rather than frozen. The loss functions for both models also look similar. At around epoch 80, the training loss continues to decrease while the validation loss starts to spike up. This could indicate overfitting, and that 80 epochs are probably enough (and will produce a better model). Nevertheless, the validation loss decreases significantly over the epochs, indicating that the model is getting better and better at predicting the correct labels.

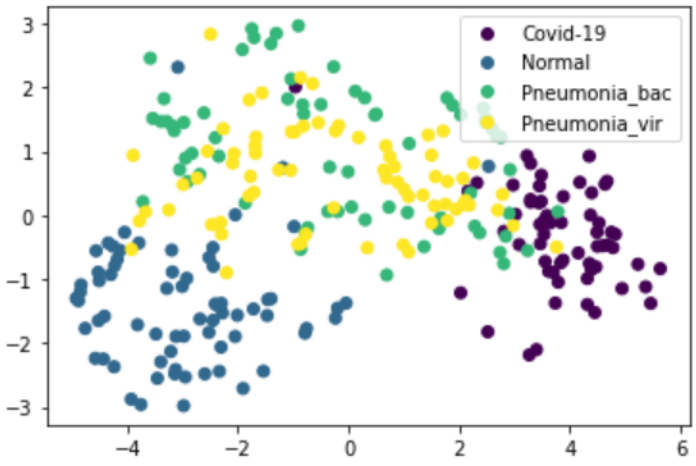
t-SNE

t-SNE plots are included below.






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



The t-SNE plot for the binary classification looks quite good. The two clusters are identifiable along a linear line and show a high degree of separation. The class assignment for the boundary points are somewhat ambiguous, but the overall plot shows that the model is able to separate the two classes clearly.
The t-SNE plot for the multiclass classification looks acceptable. We had to adjust several hyperparameters to get a good reduction on the given dataset. The parameters are included in the coding section. We found slower learning rate and more iterations help with the separation. Overall, the two plots both produce rather distinct clusters, with the Xception one producing a better result than the VGG19.

CPU vs GPU

Using Task1 as an example, Using GPU (V100) took approximately 115 seconds to train (CPU has 16 cores, using the SCC), while using CPU alone took approximately 787 seconds (on my local laptop, 8 core CPU). Even though the CPU specs don't match, the difference is still very significant. GPU runs a lot faster than CPU.
GPU Screenshot:

[5 points] Train Model

In [7]:

```
#FIT MODEL
import time

print(len(train_batches))
print(len(valid_batches))

STEP_SIZE_TRAIN=train_batches.n//train_batches.batch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

start_time = time.time()
fitted = model.fit_generator(train_batches, steps_per_epoch=STEP_SIZE_TRAIN, validation_data=valid_batches, vali
print("--- %s seconds ---" % (time.time() - start_time))
```

```
Epoch 35/40
10/10 [==============================] - 3s 262ms/step - loss: 0.6935 - acc: 0.5106 - val_loss: 0.6889 - val_acc:
0.5500
Epoch 36/40
10/10 [==============================] - 3s 258ms/step - loss: 0.6891 - acc: 0.5532 - val_loss: 0.6888 - val_acc:
0.5500
Epoch 37/40
10/10 [==============================] - 3s 274ms/step - loss: 0.6927 - acc: 0.5213 - val_loss: 0.6952 - val_acc:
0.5000
Epoch 38/40
10/10 [==============================] - 3s 268ms/step - loss: 0.6930 - acc: 0.5106 - val_loss: 0.6888 - val_acc:
0.5500
Epoch 39/40
10/10 [==============================] - 3s 265ms/step - loss: 0.6968 - acc: 0.4894 - val_loss: 0.6950 - val_acc:
0.5000
Epoch 40/40
10/10 [==============================] - 3s 263ms/step - loss: 0.6928 - acc: 0.5213 - val_loss: 0.6889 - val_acc:
0.5500
--- 115.69729375839233 seconds ---
```

CPUScreenshot:

[5 points] Train Model

In [5]:

```
#FIT MODEL
import time

print(len(train_batches))
print(len(valid_batches))

STEP_SIZE_TRAIN=train_batches.n//train_batches.batch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

start_time = time.time()
fitted = model.fit_generator(train_batches, steps_per_epoch=STEP_SIZE_TRAIN, validation_data=valid_batches, validati
print("--- %s seconds ---" % (time.time() - start_time))
```

```
Epoch 35/40
10/10 [==============================] - 19s 2s/step - loss: 0.2692 - acc: 0.9149 - val_loss: 0.1840 - val_acc:
0.9000
Epoch 36/40
10/10 [==============================] - 20s 2s/step - loss: 0.1407 - acc: 0.9468 - val_loss: 0.1258 - val_acc:
0.9500
Epoch 37/40
10/10 [==============================] - 20s 2s/step - loss: 0.1539 - acc: 0.9574 - val_loss: 0.3391 - val_acc:
0.9500
Epoch 38/40
10/10 [==============================] - 19s 2s/step - loss: 0.1323 - acc: 0.9574 - val_loss: 0.0676 - val_acc:
0.9500
Epoch 39/40
10/10 [==============================] - 19s 2s/step - loss: 0.1496 - acc: 0.9255 - val_loss: 0.0947 - val_acc:
0.9500
Epoch 40/40
10/10 [==============================] - 20s 2s/step - loss: 0.1423 - acc: 0.9400 - val_loss: 0.0250 - val_acc:
1.0000
--- 787.9125962257385 seconds ---
```

task1_template

December 7, 2021

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

2 Task 1 [Total points: 30]

2.1 Setup

- This assignment involves the following packages: 'matplotlib', 'numpy', and 'sklearn'.
- If you are using conda, use the following commands to install the above packages:

```
conda install matplotlib
conda install numpy
conda install -c anaconda scikit-learn
```

- If you are using pip, use use the following commands to install the above packages:

```
pip install matplotlib
pip install numpy
pip install sklearn
```

2.2 Data

Please download the data using the following link: COVID-19.

- After downloading 'Covid_Data_GradientCrescent.zip', unzip the file and you should see the following data structure:

|–all |———train |———test |–two |———train |———test

- Put the 'all' folder, the 'two' folder and this python notebook in the **same directory** so that the following code can correctly locate the data.

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

[1]:
```
import os

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
os.environ['OMP_NUM_THREADS'] = '1'
os.environ['CUDA_VISIBLE_DEVICES'] = '1'
tf.__version__
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
Num GPUs Available:  1
```

Load Image Data

[3]:
```
DATA_LIST = os.listdir('two/train')
DATASET_PATH  = 'two/train'
TEST_DIR =  'two/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 10  # try reducing batch size or freeze more layers if your GPU␣
 ↪runs out of memory
NUM_EPOCHS    = 40
LEARNING_RATE = 0.0005 # start off with high rate first 0.001 and experiment␣
 ↪with reducing it gradually
```

Generate Training and Validation Batches

[4]:
```
train_datagen = ImageDataGenerator(rescale=1./
 ↪255,rotation_range=50,featurewise_center = True,
                                    featurewise_std_normalization =␣
 ↪True,width_shift_range=0.2,
                                    height_shift_range=0.2,shear_range=0.
 ↪25,zoom_range=0.1,
                                    zca_whitening = True,channel_shift_range = 20,
                                    horizontal_flip = True,vertical_flip = True,
                                    validation_split = 0.2,fill_mode='constant')

train_batches = train_datagen.
 ↪flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                    ␣
 ↪shuffle=True,batch_size=BATCH_SIZE,
                                                     subset = "training",seed=42,
                                                     class_mode="binary")

valid_batches = train_datagen.
 ↪flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                    ␣
 ↪shuffle=True,batch_size=BATCH_SIZE,
                                                     subset = "validation",seed=42,
                                                     class_mode="binary")
```

```
Found 104 images belonging to 2 classes.
Found 26 images belonging to 2 classes.
```

```
/share/pkg.7/tensorflow/2.3.1/install/lib/SCC/../python3.8/site-
packages/keras_preprocessing/image/image_data_generator.py:342: UserWarning:
This ImageDataGenerator specifies `zca_whitening` which overrides setting
of`featurewise_std_normalization`.
  warnings.warn('This ImageDataGenerator specifies '
```

[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.

[30]:
```
# Note:
# https://www.tensorflow.org/guide/keras/sequential_model
# tensorflow sequential models and transfer learning

from tensorflow import keras

base_model = keras.applications.VGG16(weights='imagenet', include_top=False,␣
 ↪input_shape=(224, 224, 3))
base_model.trainable = False
'''
fine_tune_at = 15
for layer in base_model.layers[:fine_tune_at]:
  layer.trainable =  False
for layer in base_model.layers:
    print(layer, layer.trainable)
'''


model = keras.Sequential([base_model,
                          keras.layers.Flatten(),
                          keras.layers.Dense(256, activation='relu',␣
 ↪name='dense_feature'),
                          keras.layers.Dense(1, activation='sigmoid')
])

# Compile
model.compile(loss='binary_crossentropy', optimizer=keras.optimizers.
 ↪Adam(lr=LEARNING_RATE), metrics=['acc'])
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_4 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               6422784
```

```
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,137,729
Trainable params: 6,423,041
Non-trainable params: 14,714,688
_________________________________________________________________
```

[5 points] Train Model

[31]:
```
#FIT MODEL
import time

print(len(train_batches))
print(len(valid_batches))

STEP_SIZE_TRAIN=train_batches.n//train_batches.batch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

start_time = time.time()
fitted = model.fit_generator(train_batches, steps_per_epoch=STEP_SIZE_TRAIN,␣
 ↪validation_data=valid_batches, validation_steps=STEP_SIZE_VALID,␣
 ↪epochs=NUM_EPOCHS)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
11
3
/share/pkg.7/tensorflow/2.3.1/install/lib/SCC/../python3.8/site-
packages/keras_preprocessing/image/image_data_generator.py:720: UserWarning:
This ImageDataGenerator specifies `featurewise_center`, but it hasn't been fit
on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/share/pkg.7/tensorflow/2.3.1/install/lib/SCC/../python3.8/site-
packages/keras_preprocessing/image/image_data_generator.py:739: UserWarning:
This ImageDataGenerator specifies `zca_whitening`, but it hasn't been fit on any
training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '

Epoch 1/40
10/10 [==============================] - 3s 293ms/step - loss: 1.6898 - acc:
0.4787 - val_loss: 0.6026 - val_acc: 0.7500
Epoch 2/40
10/10 [==============================] - 3s 270ms/step - loss: 0.6922 - acc:
0.6400 - val_loss: 0.5811 - val_acc: 0.5500
Epoch 3/40
10/10 [==============================] - 3s 261ms/step - loss: 0.5460 - acc:
0.6596 - val_loss: 0.4199 - val_acc: 0.6500
Epoch 4/40
10/10 [==============================] - 2s 238ms/step - loss: 0.3639 - acc:
```

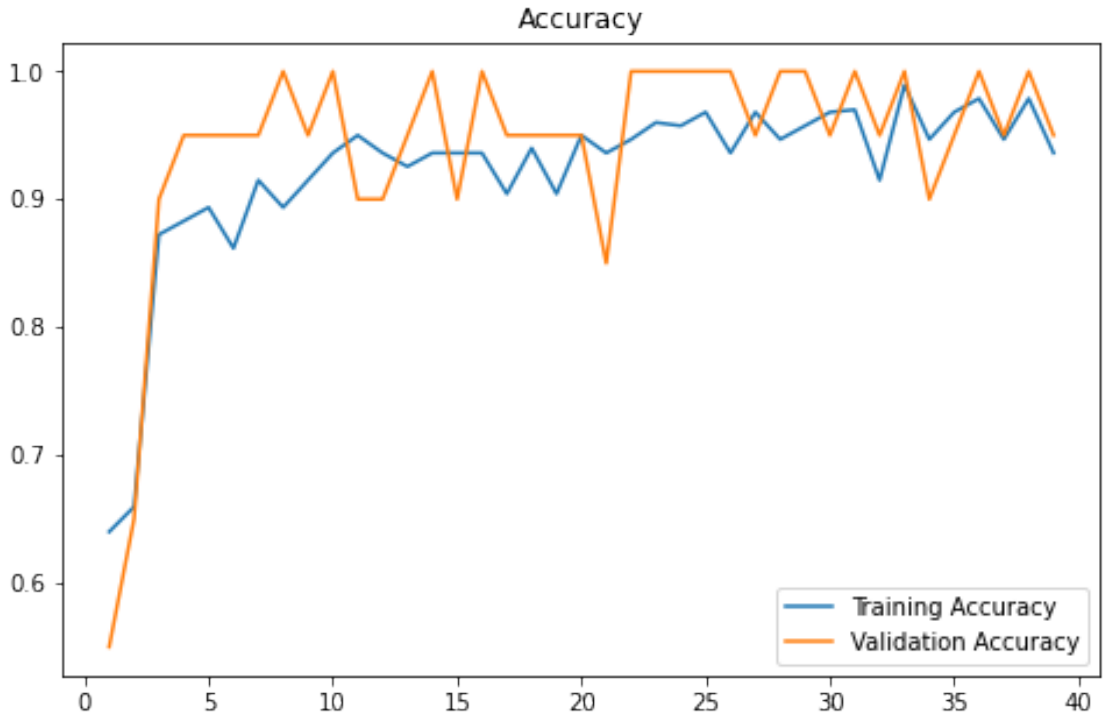
```
0.8723 - val_loss: 0.2812 - val_acc: 0.9000
Epoch 5/40
10/10 [==============================] - 3s 254ms/step - loss: 0.3415 - acc:
0.8830 - val_loss: 0.2388 - val_acc: 0.9500
Epoch 6/40
10/10 [==============================] - 2s 234ms/step - loss: 0.3116 - acc:
0.8936 - val_loss: 0.2014 - val_acc: 0.9500
Epoch 7/40
10/10 [==============================] - 3s 260ms/step - loss: 0.2937 - acc:
0.8617 - val_loss: 0.2109 - val_acc: 0.9500
Epoch 8/40
10/10 [==============================] - 3s 263ms/step - loss: 0.2649 - acc:
0.9149 - val_loss: 0.1869 - val_acc: 0.9500
Epoch 9/40
10/10 [==============================] - 3s 250ms/step - loss: 0.2179 - acc:
0.8936 - val_loss: 0.1025 - val_acc: 1.0000
Epoch 10/40
10/10 [==============================] - 2s 245ms/step - loss: 0.2238 - acc:
0.9149 - val_loss: 0.1278 - val_acc: 0.9500
Epoch 11/40
10/10 [==============================] - 3s 262ms/step - loss: 0.1827 - acc:
0.9362 - val_loss: 0.1438 - val_acc: 1.0000
Epoch 12/40
10/10 [==============================] - 3s 274ms/step - loss: 0.1712 - acc:
0.9500 - val_loss: 0.2565 - val_acc: 0.9000
Epoch 13/40
10/10 [==============================] - 3s 255ms/step - loss: 0.1954 - acc:
0.9362 - val_loss: 0.1999 - val_acc: 0.9000
Epoch 14/40
10/10 [==============================] - 3s 256ms/step - loss: 0.1828 - acc:
0.9255 - val_loss: 0.1961 - val_acc: 0.9500
Epoch 15/40
10/10 [==============================] - 3s 253ms/step - loss: 0.1807 - acc:
0.9362 - val_loss: 0.0598 - val_acc: 1.0000
Epoch 16/40
10/10 [==============================] - 3s 265ms/step - loss: 0.1717 - acc:
0.9362 - val_loss: 0.1634 - val_acc: 0.9000
Epoch 17/40
10/10 [==============================] - 3s 268ms/step - loss: 0.1858 - acc:
0.9362 - val_loss: 0.0398 - val_acc: 1.0000
Epoch 18/40
10/10 [==============================] - 2s 240ms/step - loss: 0.2024 - acc:
0.9043 - val_loss: 0.0966 - val_acc: 0.9500
Epoch 19/40
10/10 [==============================] - 3s 264ms/step - loss: 0.2185 - acc:
0.9400 - val_loss: 0.1909 - val_acc: 0.9500
Epoch 20/40
10/10 [==============================] - 3s 254ms/step - loss: 0.1964 - acc:
```

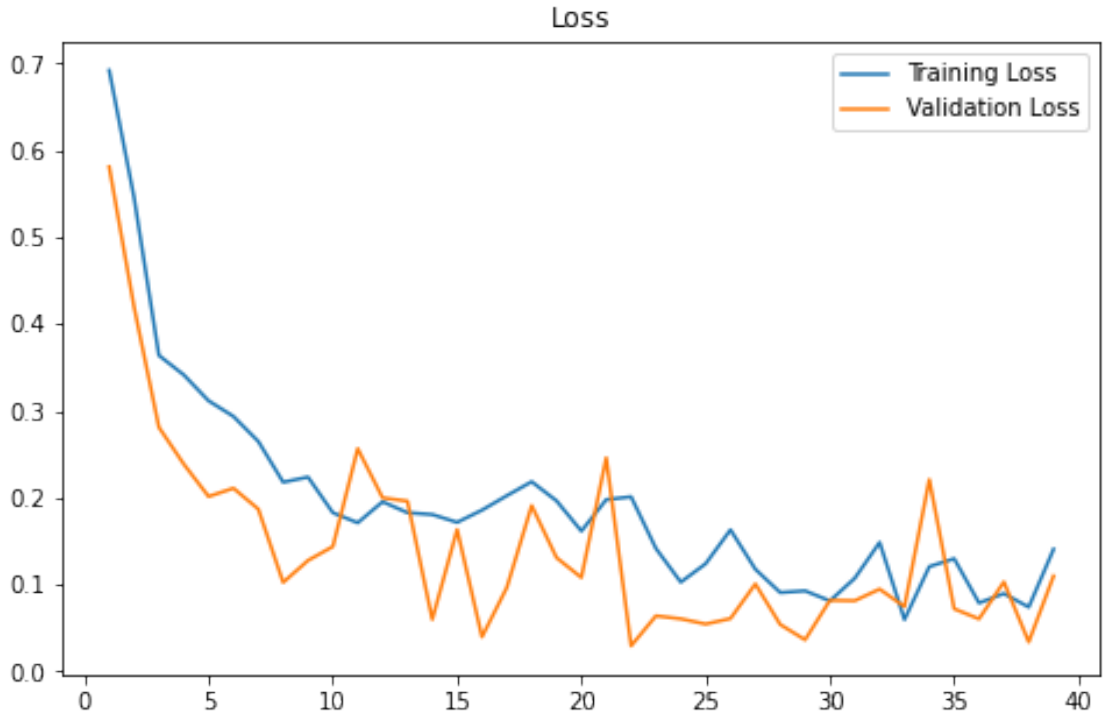
```
0.9043 - val_loss: 0.1310 - val_acc: 0.9500
Epoch 21/40
10/10 [==============================] - 3s 263ms/step - loss: 0.1616 - acc:
0.9500 - val_loss: 0.1081 - val_acc: 0.9500
Epoch 22/40
10/10 [==============================] - 3s 251ms/step - loss: 0.1978 - acc:
0.9362 - val_loss: 0.2460 - val_acc: 0.8500
Epoch 23/40
10/10 [==============================] - 3s 255ms/step - loss: 0.2011 - acc:
0.9468 - val_loss: 0.0294 - val_acc: 1.0000
Epoch 24/40
10/10 [==============================] - 3s 263ms/step - loss: 0.1416 - acc:
0.9600 - val_loss: 0.0640 - val_acc: 1.0000
Epoch 25/40
10/10 [==============================] - 3s 255ms/step - loss: 0.1028 - acc:
0.9574 - val_loss: 0.0606 - val_acc: 1.0000
Epoch 26/40
10/10 [==============================] - 3s 264ms/step - loss: 0.1240 - acc:
0.9681 - val_loss: 0.0545 - val_acc: 1.0000
Epoch 27/40
10/10 [==============================] - 3s 260ms/step - loss: 0.1631 - acc:
0.9362 - val_loss: 0.0609 - val_acc: 1.0000
Epoch 28/40
10/10 [==============================] - 3s 272ms/step - loss: 0.1177 - acc:
0.9681 - val_loss: 0.1008 - val_acc: 0.9500
Epoch 29/40
10/10 [==============================] - 3s 277ms/step - loss: 0.0908 - acc:
0.9468 - val_loss: 0.0540 - val_acc: 1.0000
Epoch 30/40
10/10 [==============================] - 3s 268ms/step - loss: 0.0928 - acc:
0.9574 - val_loss: 0.0364 - val_acc: 1.0000
Epoch 31/40
10/10 [==============================] - 2s 247ms/step - loss: 0.0813 - acc:
0.9681 - val_loss: 0.0819 - val_acc: 0.9500
Epoch 32/40
10/10 [==============================] - 3s 276ms/step - loss: 0.1073 - acc:
0.9700 - val_loss: 0.0814 - val_acc: 1.0000
Epoch 33/40
10/10 [==============================] - 3s 270ms/step - loss: 0.1486 - acc:
0.9149 - val_loss: 0.0948 - val_acc: 0.9500
Epoch 34/40
10/10 [==============================] - 2s 250ms/step - loss: 0.0597 - acc:
0.9894 - val_loss: 0.0746 - val_acc: 1.0000
Epoch 35/40
10/10 [==============================] - 3s 253ms/step - loss: 0.1207 - acc:
0.9468 - val_loss: 0.2210 - val_acc: 0.9000
Epoch 36/40
10/10 [==============================] - 3s 261ms/step - loss: 0.1298 - acc:
```

```
0.9681 - val_loss: 0.0721 - val_acc: 0.9500
Epoch 37/40
10/10 [==============================] - 3s 257ms/step - loss: 0.0788 - acc:
0.9787 - val_loss: 0.0605 - val_acc: 1.0000
Epoch 38/40
10/10 [==============================] - 3s 267ms/step - loss: 0.0896 - acc:
0.9468 - val_loss: 0.1031 - val_acc: 0.9500
Epoch 39/40
10/10 [==============================] - 2s 235ms/step - loss: 0.0741 - acc:
0.9787 - val_loss: 0.0339 - val_acc: 1.0000
Epoch 40/40
10/10 [==============================] - 3s 270ms/step - loss: 0.1408 - acc:
0.9362 - val_loss: 0.1098 - val_acc: 0.9500
--- 114.68965125083923 seconds ---
```

[5 points] Plot Accuracy and Loss During Training

[32]:
```
import matplotlib.pyplot as plt

acc = fitted.history['acc']
loss = fitted.history['loss']
val_acc = fitted.history['val_acc']
val_loss = fitted.history['val_loss']

# accuracy plot
plt.figure(figsize=(8, 5))
plt.plot(np.arange(1,NUM_EPOCHS), acc[1:], label='Training Accuracy')
plt.plot(np.arange(1,NUM_EPOCHS), val_acc[1:], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()
plt.show()

# loss plot
plt.figure(figsize=(8, 5))
plt.plot(np.arange(1,NUM_EPOCHS), loss[1:], label='Training Loss')
plt.plot(np.arange(1,NUM_EPOCHS), val_loss[1:], label='Validation Loss')
plt.title('Loss')
plt.legend()
plt.show()
```

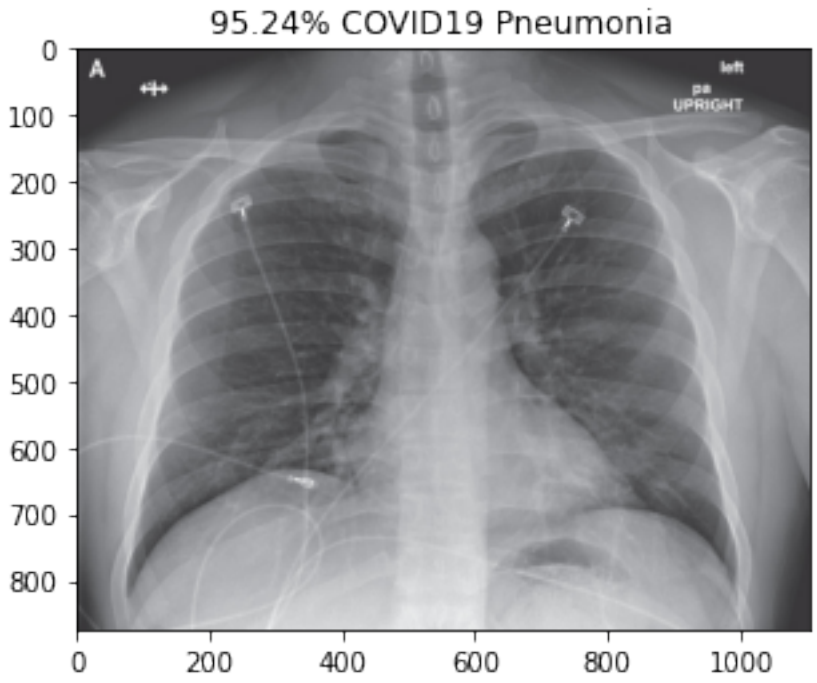

Accuracy
1.0
0.9
0.8
0.7
0.6
0
5
10
15
20
25
30
35
40
Training Accuracy
Validation Accuracy


Loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
30
35
40
Training Loss
Validation Loss

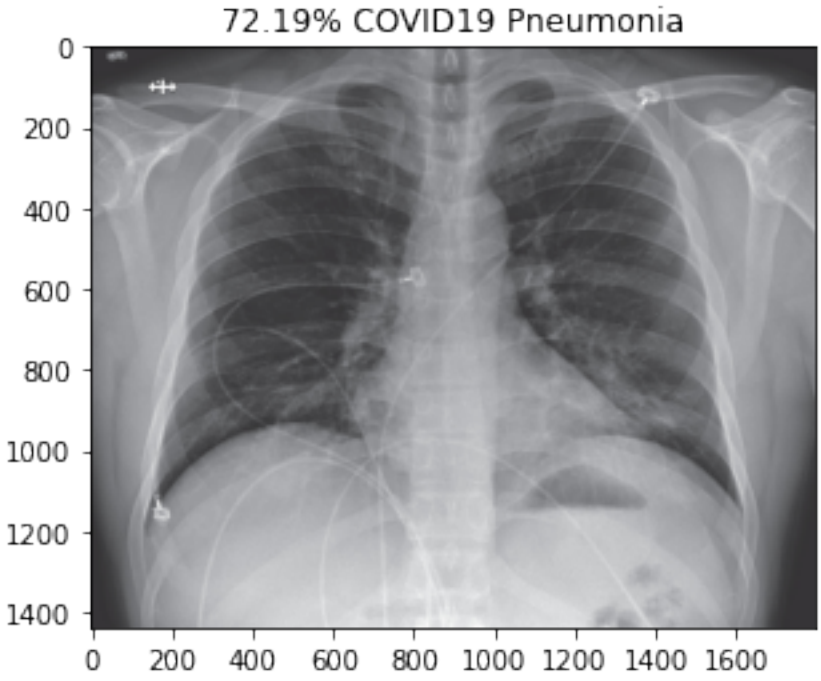
Plot Test Results

[33]:
```
import matplotlib.image as mpimg

test_datagen = ImageDataGenerator(rescale=1. / 255)
eval_generator = test_datagen.
 ↪flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                    ␣
 ↪batch_size=1,shuffle=True,seed=42,class_mode="binary")
eval_generator.reset()
pred = model.predict_generator(eval_generator,18,verbose=1)
for index, probability in enumerate(pred):
    image_path = TEST_DIR + "/" +eval_generator.filenames[index]
    image = mpimg.imread(image_path)
    if image.ndim < 3:
        image = np.reshape(image,(image.shape[0],image.shape[1],1))
        image = np.concatenate([image, image, image], 2)
#         print(image.shape)

    pixels = np.array(image)
    plt.imshow(pixels)

    print(eval_generator.filenames[index])
    if probability > 0.5:
        plt.title("%.2f" % (probability[0]*100) + "% Normal")
    else:
        plt.title("%.2f" % ((1-probability[0])*100) + "% COVID19 Pneumonia")
    plt.show()
```

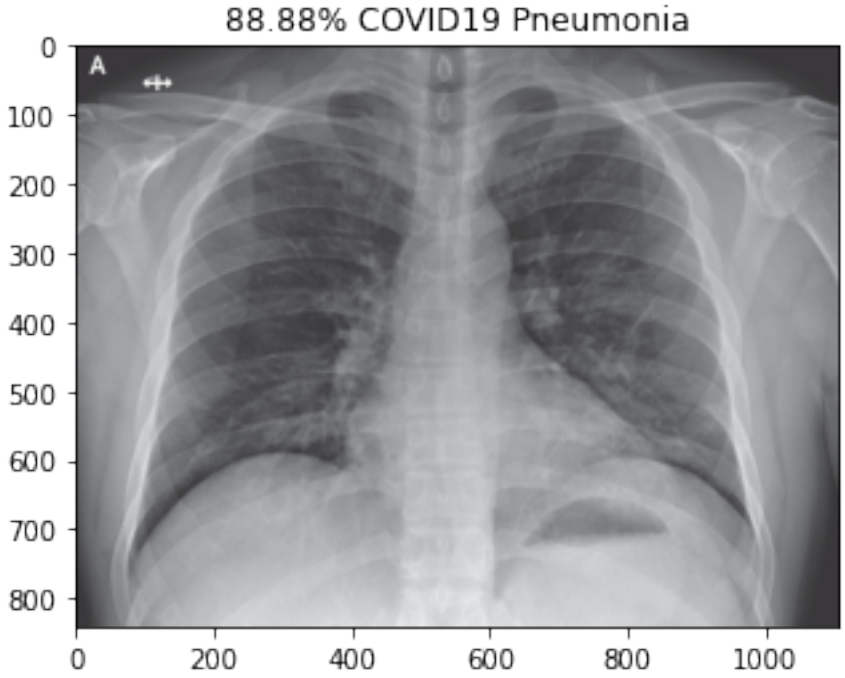
```
Found 18 images belonging to 2 classes.
18/18 [==============================] - 0s 19ms/step
covid/nejmoa2001191_f3-PA.jpeg
```



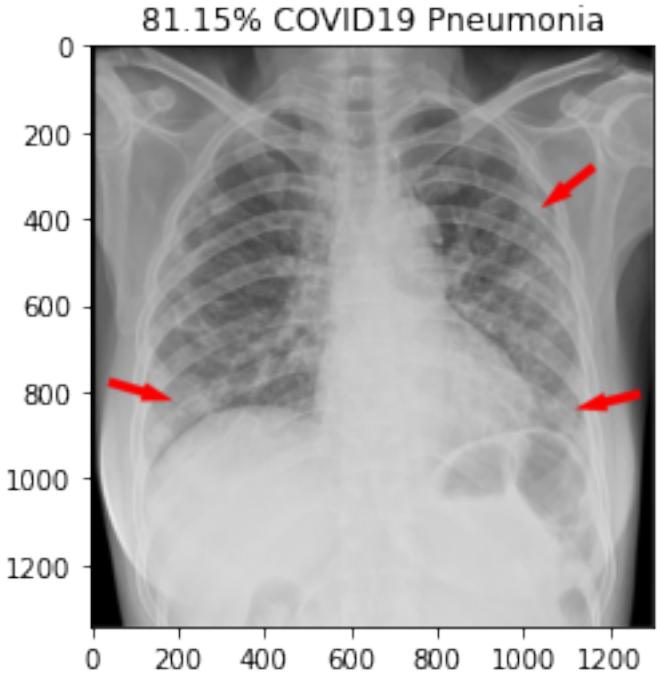
covid/nejmoa2001191_f4.jpeg



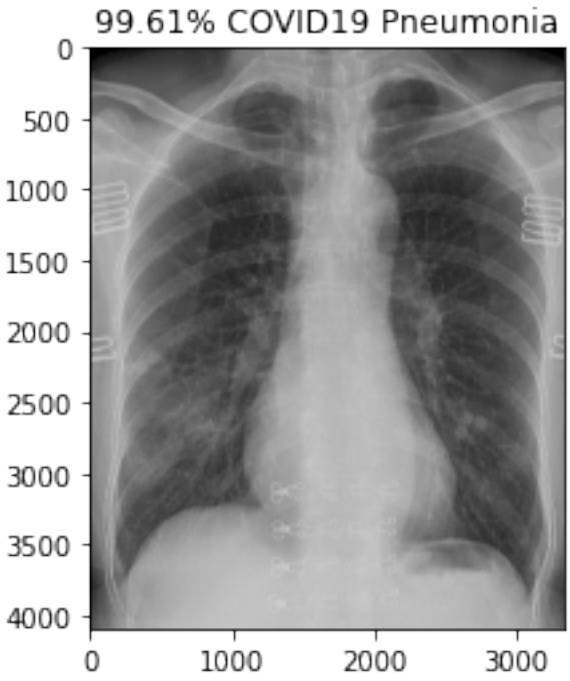
covid/nejmoa2001191_f5-PA.jpeg



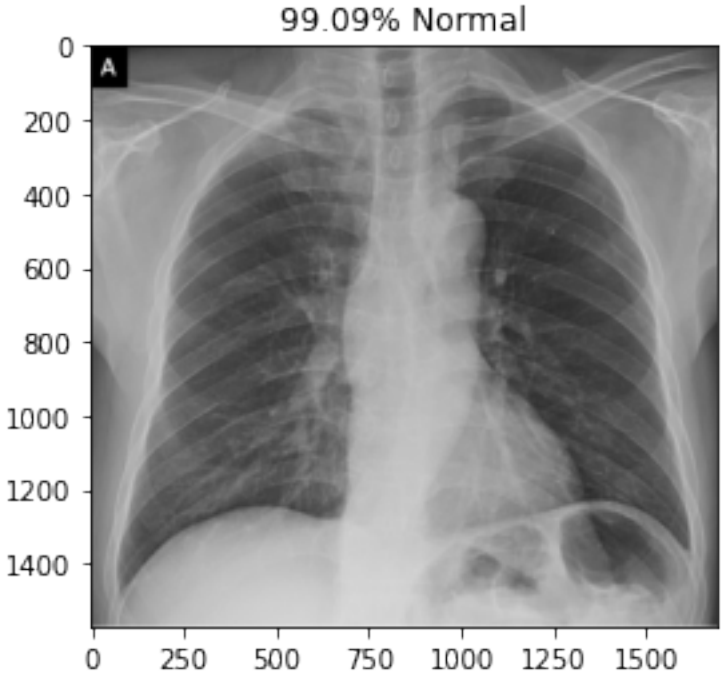
covid/radiol.2020200490.fig3.jpeg



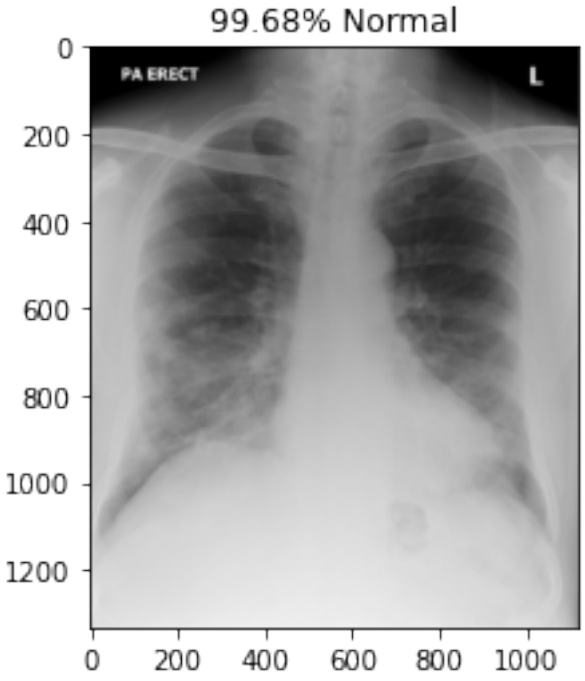
covid/ryct.2020200028.fig1a.jpeg



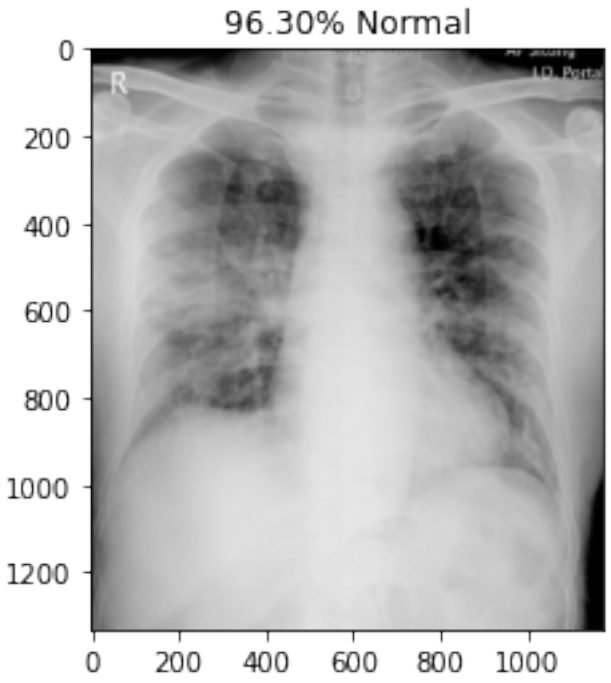
covid/ryct.2020200034.fig2.jpeg



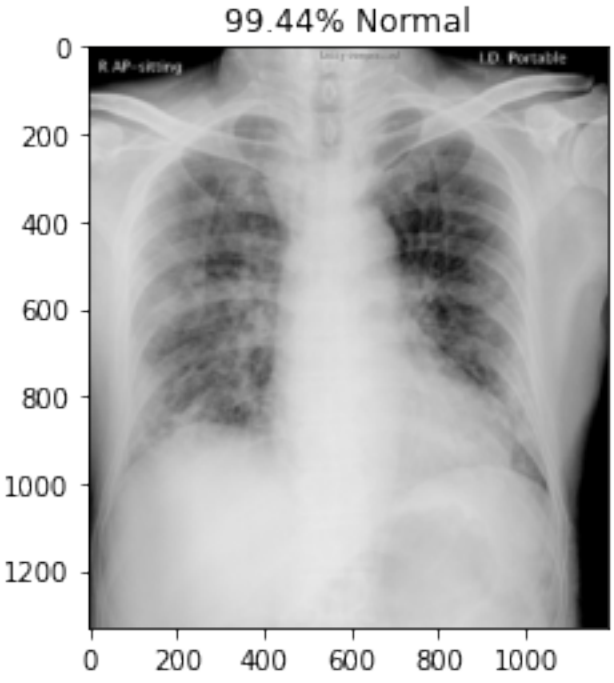
covid/ryct.2020200034.fig5-day0.jpeg



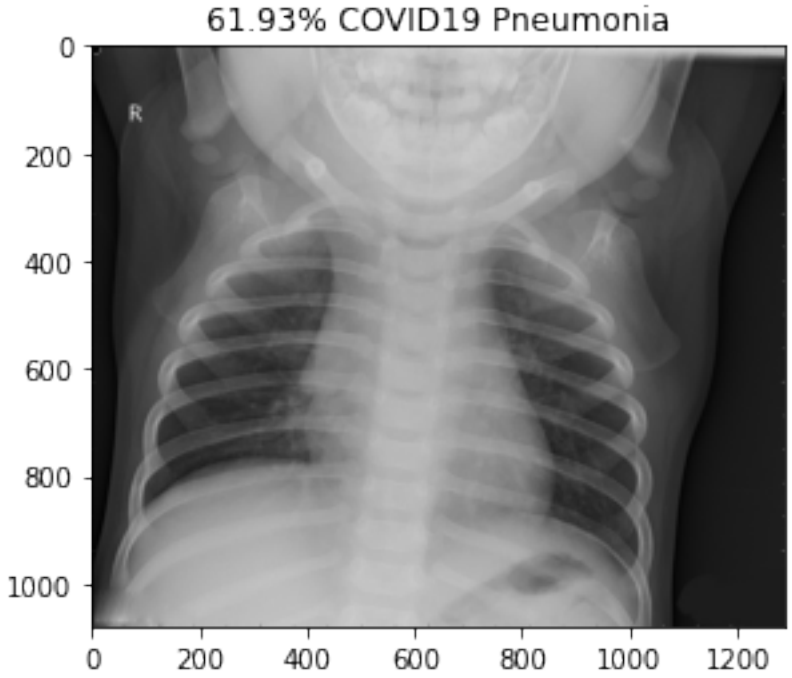
covid/ryct.2020200034.fig5-day4.jpeg



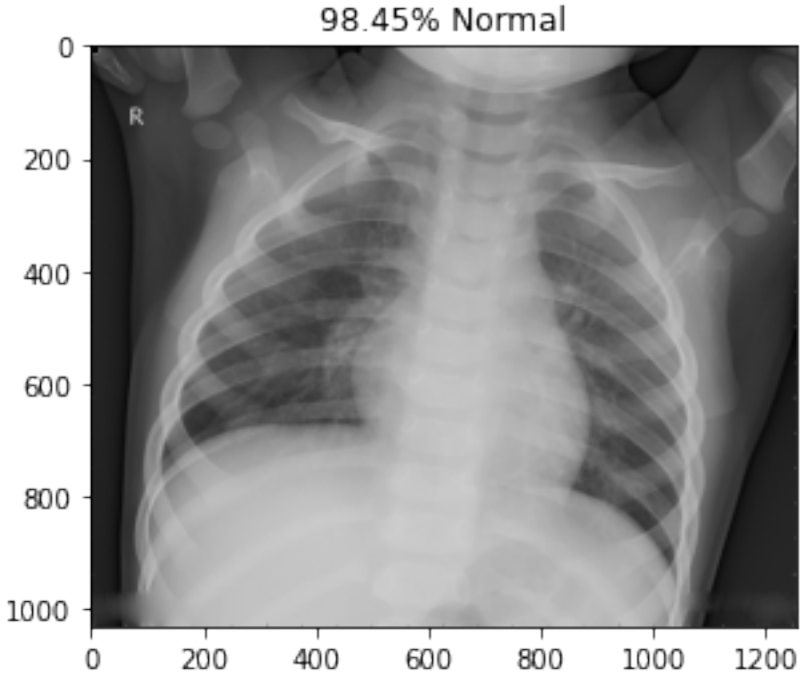
covid/ryct.2020200034.fig5-day7.jpeg



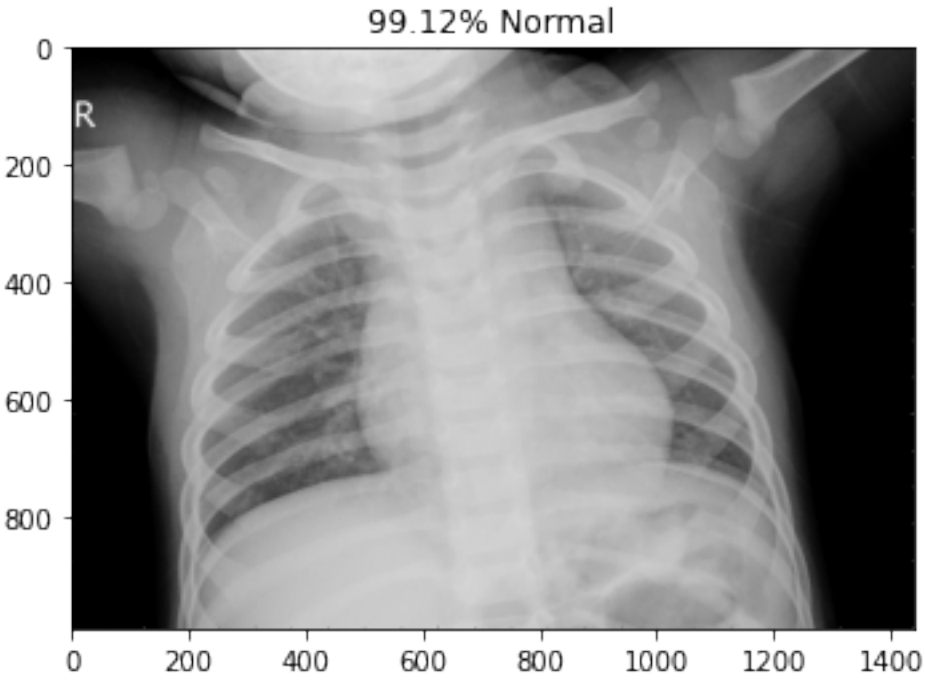
normal/NORMAL2-IM-1385-0001.jpeg



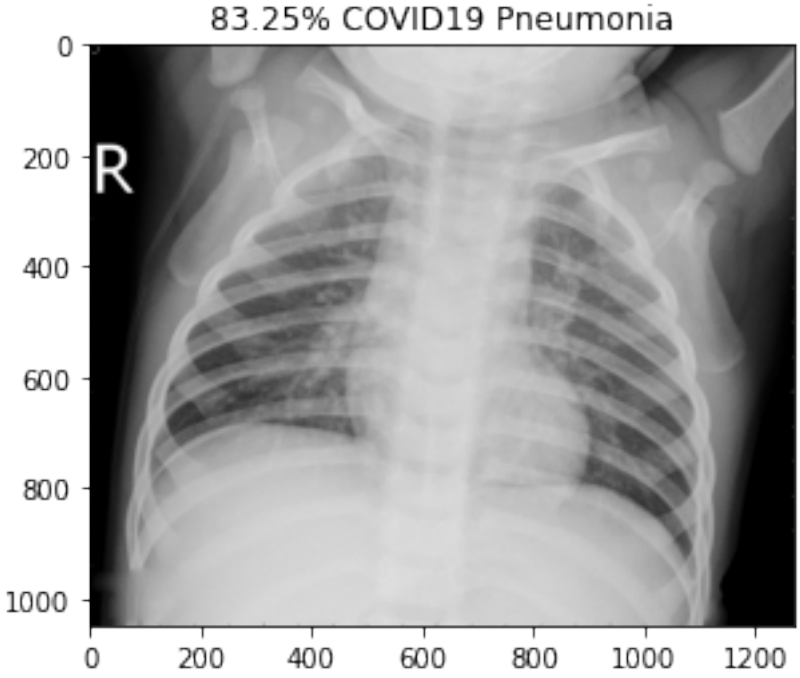
normal/NORMAL2-IM-1396-0001.jpeg



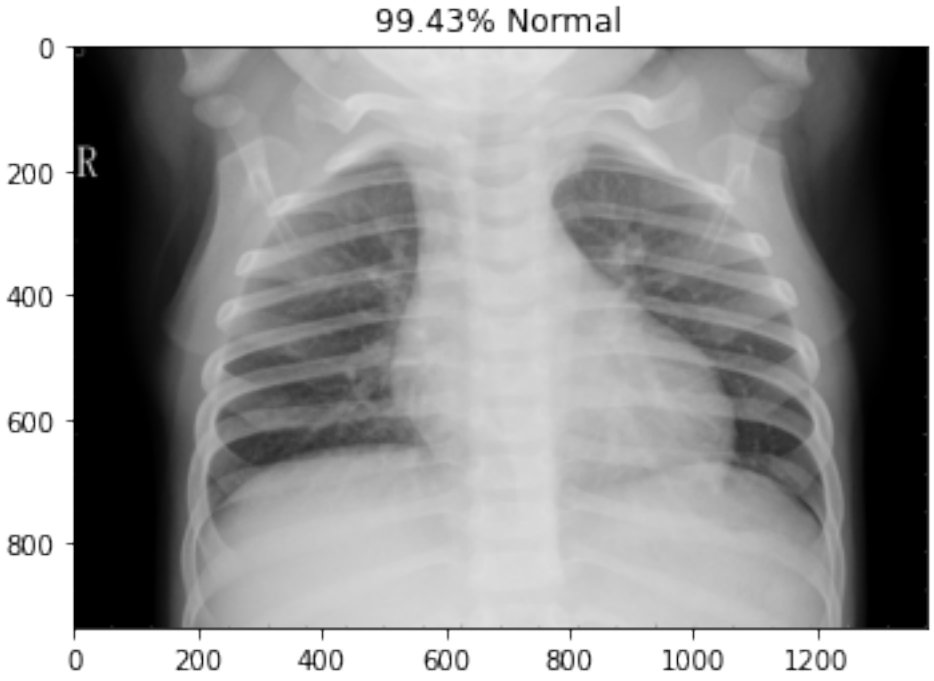
normal/NORMAL2-IM-1400-0001.jpeg



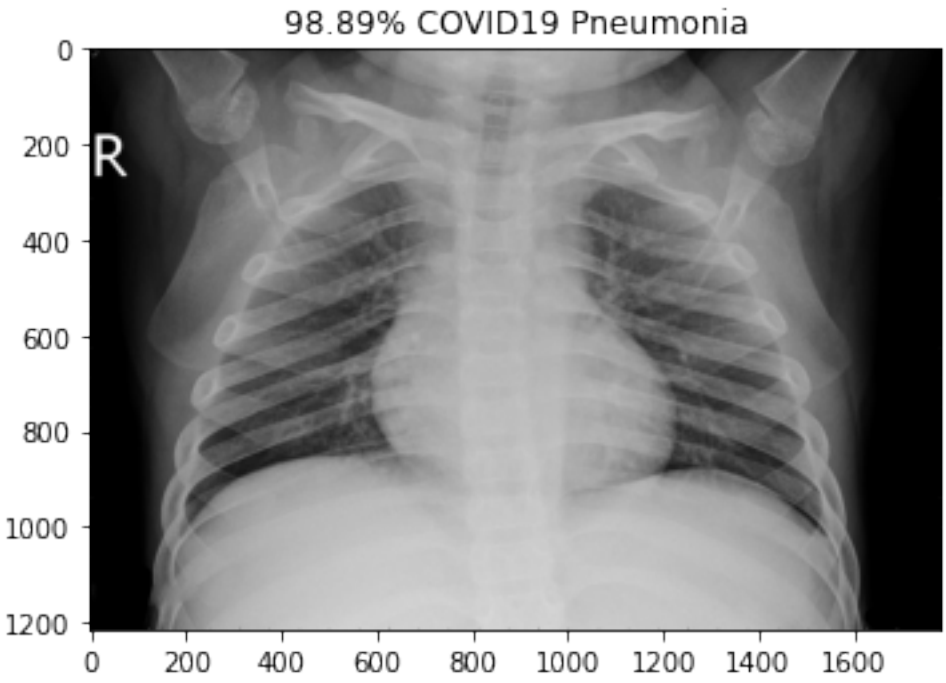
normal/NORMAL2-IM-1401-0001.jpeg



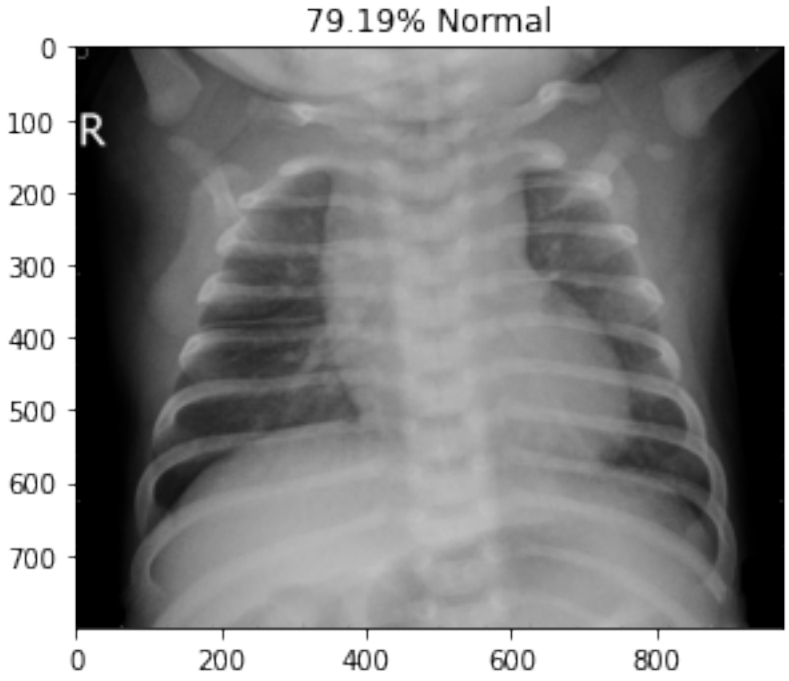
normal/NORMAL2-IM-1406-0001.jpeg



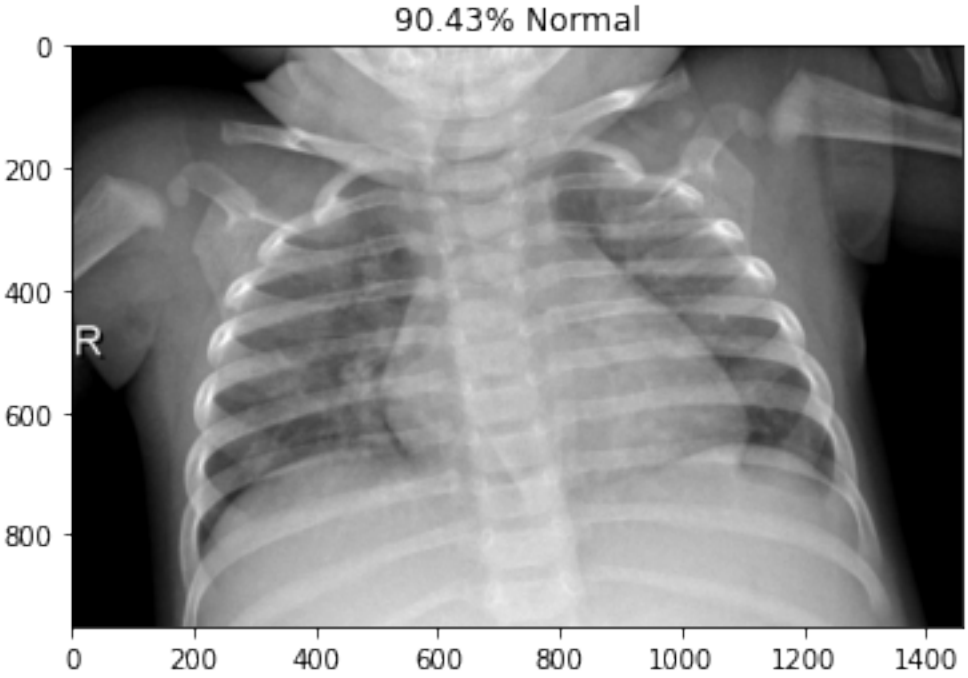
normal/NORMAL2-IM-1412-0001.jpeg



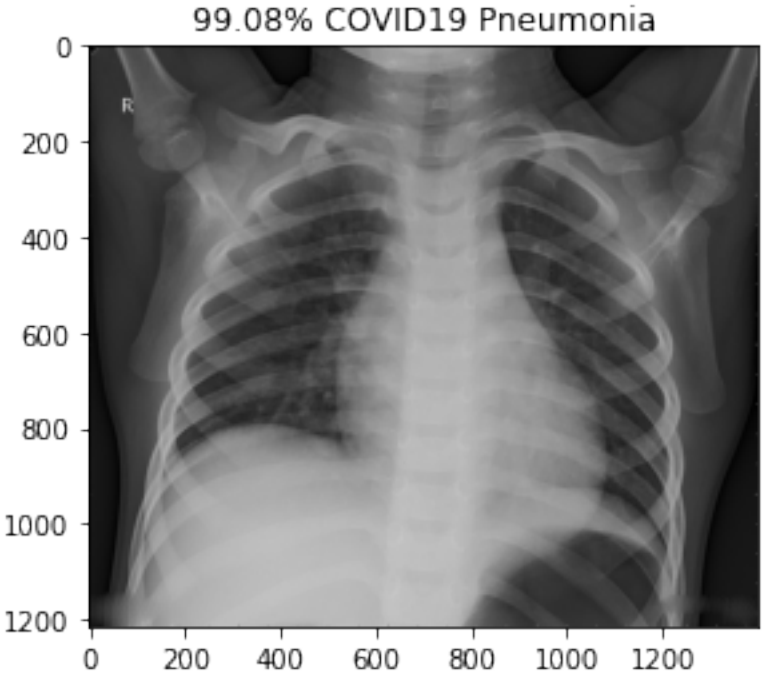
`normal/NORMAL2-IM-1419-0001.jpeg`



normal/NORMAL2-IM-1422-0001.jpeg



normal/NORMAL2-IM-1423-0001.jpeg

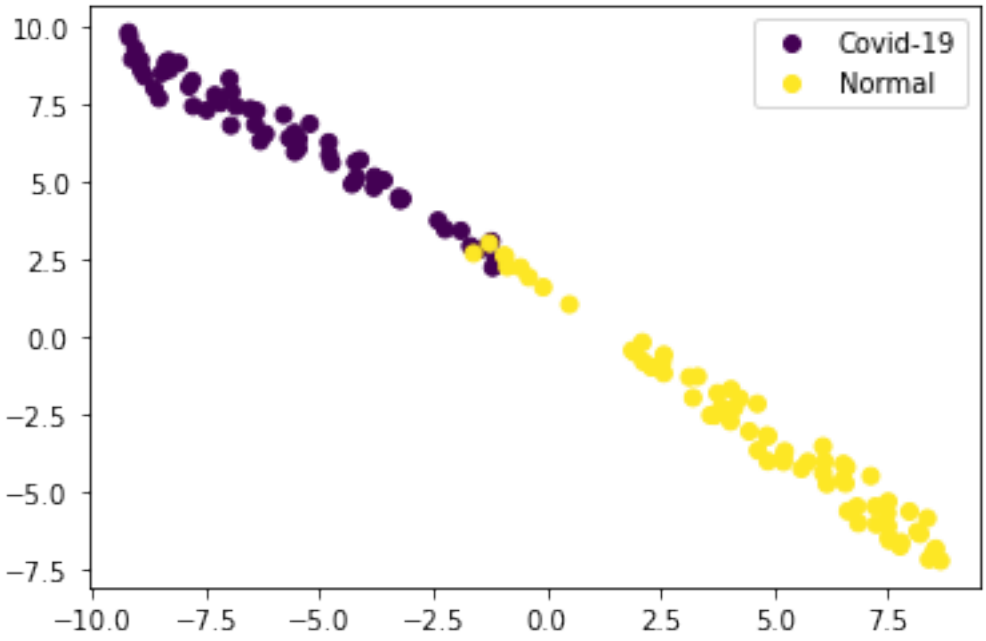


2.4 [10 points] TSNE Plot

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely used technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. After training is complete, extract features from a specific deep layer of your choice, use t-SNE to reduce the dimensionality of your extracted features to 2 dimensions and plot the resulting 2D features.

[34]:
```
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.Model(inputs=model.input,
                                          outputs=model.get_layer('dense_feature').
 ↪output)
tsne_data_generator = test_datagen.
 ↪flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                                 ␣
 ↪batch_size=1,shuffle=False,seed=42,class_mode="binary")
y_true = tsne_data_generator.classes

# Compile
intermediate_layer_model.compile(loss='binary_crossentropy', optimizer=keras.
 ↪optimizers.Adam(lr=LEARNING_RATE), metrics=['acc'])

features = intermediate_layer_model.predict(tsne_data_generator)

tsne = TSNE(n_components=2).fit_transform(features)

my_label = ['Covid-19', 'Normal']

scatter = plt.scatter(tsne[:,0], tsne[:,1], c=y_true)
handles, _ = scatter.legend_elements(prop='colors')
plt.legend(handles, my_label)
plt.show()
```

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


Covid-19
Normal
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

task2_template

December 7, 2021

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

2 Task 2 [Total points: 30]

2.1 Setup

- This assignment involves the following packages: 'matplotlib', 'numpy', and 'sklearn'.
- If you are using conda, use the following commands to install the above packages:

```
conda install matplotlib
conda install numpy
conda install -c anaconda scikit-learn
```

- If you are using pip, use use the following commands to install the above packages:

```
pip install matplotlib
pip install numpy
pip install sklearn
```

2.2 Data

Please download the data using the following link: COVID-19.

- After downloading 'Covid_Data_GradientCrescent.zip', unzip the file and you should see the following data structure:

|–all |———train |———test |–two |———train |———test

- Put the 'all' folder, the 'two' folder and this python notebook in the **same directory** so that the following code can correctly locate the data.

2.3 [20 points] Multi-class Classification

[20]:
```
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__
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
Num GPUs Available:  1
```

Load Image Data

[21]:
```
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␣
↪runs out of memory
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experiment␣
↪with reducing it gradually
```

Generate Training and Validation Batches

[22]:
```
train_datagen = ImageDataGenerator(rescale=1./
↪255,rotation_range=50,featurewise_center = True,
                                   featurewise_std_normalization =␣
↪True,width_shift_range=0.2,
                                   height_shift_range=0.2,shear_range=0.
↪25,zoom_range=0.1,
                                   zca_whitening = True,channel_shift_range = 20,
                                   horizontal_flip = True,vertical_flip = True,
                                   validation_split = 0.2,fill_mode='constant')


train_batches = train_datagen.
↪flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                    ␣
↪shuffle=True,batch_size=BATCH_SIZE,
                                                     subset = "training",seed=42,
                                                     class_mode="categorical")

valid_batches = train_datagen.
↪flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                    ␣
↪shuffle=True,batch_size=BATCH_SIZE,
                                                     subset = "validation",
                                                    ␣
↪seed=42,class_mode="categorical")
```

```
Found 216 images belonging to 4 classes.
Found 54 images belonging to 4 classes.
```

```
/share/pkg.7/tensorflow/2.3.1/install/lib/SCC/../python3.8/site-
packages/keras_preprocessing/image/image_data_generator.py:342: UserWarning:
This ImageDataGenerator specifies `zca_whitening` which overrides setting
of`featurewise_std_normalization`.
  warnings.warn('This ImageDataGenerator specifies '
```

[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.

[23]:
```
from tensorflow import keras

# VGG19
base_model = keras.applications.VGG19(weights='imagenet', include_top=False,␣
 ↪input_shape=(224, 224, 3))
base_model.trainable = False
model = keras.Sequential([base_model,
                          keras.layers.Flatten(),
                          keras.layers.Dense(256, activation='relu',␣
 ↪name='dense_feature'),
                          keras.layers.Dropout(rate=0.2),
                          keras.layers.Dense(4, activation='softmax')
])


# Xception
base_model2 = keras.applications.Xception(weights='imagenet', include_top=False,␣
 ↪input_shape=(224, 224, 3))
base_model2.trainable = False
model2 = keras.Sequential([base_model2,
                          keras.layers.Flatten(),
                          keras.layers.Dense(256, activation='relu',␣
 ↪name='dense_feature'),
                           keras.layers.Dropout(rate=0.2),
                           keras.layers.Dense(4, activation='softmax')
])

# Compile
model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.
 ↪Adam(lr=LEARNING_RATE), metrics=['acc'])
model.summary()

model2.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.
 ↪Adam(lr=LEARNING_RATE), metrics=['acc'])
model2.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80142336/80134624 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83689472/83683744 [==============================] - 1s 0us/step
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 7, 7, 512)         20024384
_________________________________________________________________
flatten_8 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               6422784
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 1028
=================================================================
Total params: 26,448,196
Trainable params: 6,423,812
Non-trainable params: 20,024,384
_________________________________________________________________
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Functional)        (None, 7, 7, 2048)        20861480
_________________________________________________________________
flatten_9 (Flatten)          (None, 100352)            0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               25690368
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 4)                 1028
=================================================================
Total params: 46,552,876
Trainable params: 25,691,396
Non-trainable params: 20,861,480
_________________________________________________________________
```

[5 points] Train Model

[24]:
```
#FIT MODEL
print(len(train_batches))
print(len(valid_batches))
```

```
STEP_SIZE_TRAIN=train_batches.n//train_batches.batch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

fitted = model.fit_generator(train_batches, steps_per_epoch=STEP_SIZE_TRAIN,␣
 ↪validation_data=valid_batches, validation_steps=STEP_SIZE_VALID,␣
 ↪epochs=NUM_EPOCHS)
fitted2 = model2.fit_generator(train_batches, steps_per_epoch=STEP_SIZE_TRAIN,␣
 ↪validation_data=valid_batches, validation_steps=STEP_SIZE_VALID,␣
 ↪epochs=NUM_EPOCHS)
```

```
22
6
Epoch 1/100
21/21 [==============================] - 5s 226ms/step - loss: 1.6315 - acc:
0.3447 - val_loss: 1.2552 - val_acc: 0.4400
Epoch 2/100
21/21 [==============================] - 4s 208ms/step - loss: 1.2782 - acc:
0.4029 - val_loss: 1.0264 - val_acc: 0.6000
Epoch 3/100
21/21 [==============================] - 4s 204ms/step - loss: 1.1825 - acc:
0.4757 - val_loss: 1.0828 - val_acc: 0.4200
Epoch 4/100
21/21 [==============================] - 4s 207ms/step - loss: 1.0749 - acc:
0.4854 - val_loss: 0.9261 - val_acc: 0.6800
Epoch 5/100
21/21 [==============================] - 4s 205ms/step - loss: 1.0542 - acc:
0.5437 - val_loss: 0.8920 - val_acc: 0.6400
Epoch 6/100
21/21 [==============================] - 4s 204ms/step - loss: 0.9247 - acc:
0.5388 - val_loss: 0.9366 - val_acc: 0.5800
Epoch 7/100
21/21 [==============================] - 4s 212ms/step - loss: 1.0150 - acc:
0.5583 - val_loss: 0.8995 - val_acc: 0.5800
Epoch 8/100
21/21 [==============================] - 4s 209ms/step - loss: 0.9431 - acc:
0.5777 - val_loss: 0.9428 - val_acc: 0.5400
Epoch 9/100
21/21 [==============================] - 4s 203ms/step - loss: 0.9152 - acc:
0.6165 - val_loss: 0.7888 - val_acc: 0.6600
Epoch 10/100
21/21 [==============================] - 4s 207ms/step - loss: 1.0014 - acc:
0.5874 - val_loss: 0.9155 - val_acc: 0.4800
Epoch 11/100
21/21 [==============================] - 4s 205ms/step - loss: 0.9127 - acc:
0.6408 - val_loss: 0.9007 - val_acc: 0.5200
Epoch 12/100
21/21 [==============================] - 4s 206ms/step - loss: 0.8531 - acc:
```

```
0.6068 - val_loss: 0.9357 - val_acc: 0.5000
Epoch 13/100
21/21 [==============================] - 4s 201ms/step - loss: 0.8135 - acc:
0.6650 - val_loss: 0.7745 - val_acc: 0.6400
Epoch 14/100
21/21 [==============================] - 4s 211ms/step - loss: 0.8895 - acc:
0.5825 - val_loss: 0.7454 - val_acc: 0.6000
Epoch 15/100
21/21 [==============================] - 4s 209ms/step - loss: 0.7587 - acc:
0.7184 - val_loss: 0.8298 - val_acc: 0.5600
Epoch 16/100
21/21 [==============================] - 4s 211ms/step - loss: 0.8429 - acc:
0.5971 - val_loss: 0.7157 - val_acc: 0.7000
Epoch 17/100
21/21 [==============================] - 4s 211ms/step - loss: 0.8465 - acc:
0.6117 - val_loss: 0.7784 - val_acc: 0.5800
Epoch 18/100
21/21 [==============================] - 4s 212ms/step - loss: 0.7750 - acc:
0.6505 - val_loss: 0.7575 - val_acc: 0.6200
Epoch 19/100
21/21 [==============================] - 4s 213ms/step - loss: 0.8494 - acc:
0.6311 - val_loss: 0.7885 - val_acc: 0.6400
Epoch 20/100
21/21 [==============================] - 4s 203ms/step - loss: 0.7825 - acc:
0.6456 - val_loss: 0.7970 - val_acc: 0.6000
Epoch 21/100
21/21 [==============================] - 4s 212ms/step - loss: 0.7525 - acc:
0.6408 - val_loss: 0.8399 - val_acc: 0.6000
Epoch 22/100
21/21 [==============================] - 4s 212ms/step - loss: 0.7986 - acc:
0.6359 - val_loss: 0.7276 - val_acc: 0.7000
Epoch 23/100
21/21 [==============================] - 4s 209ms/step - loss: 0.8389 - acc:
0.6311 - val_loss: 0.8148 - val_acc: 0.5800
Epoch 24/100
21/21 [==============================] - 4s 212ms/step - loss: 0.7614 - acc:
0.6602 - val_loss: 0.8472 - val_acc: 0.6200
Epoch 25/100
21/21 [==============================] - 4s 205ms/step - loss: 0.7793 - acc:
0.6845 - val_loss: 0.6997 - val_acc: 0.6400
Epoch 26/100
21/21 [==============================] - 4s 210ms/step - loss: 0.7361 - acc:
0.6893 - val_loss: 0.6831 - val_acc: 0.6600
Epoch 27/100
21/21 [==============================] - 4s 209ms/step - loss: 0.7057 - acc:
0.7282 - val_loss: 0.7133 - val_acc: 0.7000
Epoch 28/100
21/21 [==============================] - 4s 214ms/step - loss: 0.7621 - acc:
```

```
0.6333 - val_loss: 0.6424 - val_acc: 0.6400
Epoch 29/100
21/21 [==============================] - 4s 209ms/step - loss: 0.8150 - acc:
0.6165 - val_loss: 0.7764 - val_acc: 0.6000
Epoch 30/100
21/21 [==============================] - 4s 196ms/step - loss: 0.7961 - acc:
0.6262 - val_loss: 0.7324 - val_acc: 0.6000
Epoch 31/100
21/21 [==============================] - 4s 206ms/step - loss: 0.7486 - acc:
0.7039 - val_loss: 0.6578 - val_acc: 0.6600
Epoch 32/100
21/21 [==============================] - 4s 209ms/step - loss: 0.6718 - acc:
0.7330 - val_loss: 0.6963 - val_acc: 0.6400
Epoch 33/100
21/21 [==============================] - 4s 212ms/step - loss: 0.6760 - acc:
0.7087 - val_loss: 0.8093 - val_acc: 0.5400
Epoch 34/100
21/21 [==============================] - 4s 208ms/step - loss: 0.7041 - acc:
0.6893 - val_loss: 0.6806 - val_acc: 0.6600
Epoch 35/100
21/21 [==============================] - 4s 204ms/step - loss: 0.7258 - acc:
0.6796 - val_loss: 0.6808 - val_acc: 0.7000
Epoch 36/100
21/21 [==============================] - 4s 203ms/step - loss: 0.7154 - acc:
0.6845 - val_loss: 0.6819 - val_acc: 0.6600
Epoch 37/100
21/21 [==============================] - 4s 207ms/step - loss: 0.6793 - acc:
0.7282 - val_loss: 0.7039 - val_acc: 0.6800
Epoch 38/100
21/21 [==============================] - 4s 210ms/step - loss: 0.7075 - acc:
0.7184 - val_loss: 0.5918 - val_acc: 0.7200
Epoch 39/100
21/21 [==============================] - 4s 205ms/step - loss: 0.6400 - acc:
0.7087 - val_loss: 0.7956 - val_acc: 0.6000
Epoch 40/100
21/21 [==============================] - 4s 210ms/step - loss: 0.7548 - acc:
0.6553 - val_loss: 0.8489 - val_acc: 0.6200
Epoch 41/100
21/21 [==============================] - 4s 197ms/step - loss: 0.6879 - acc:
0.6990 - val_loss: 0.7109 - val_acc: 0.6400
Epoch 42/100
21/21 [==============================] - 4s 209ms/step - loss: 0.7989 - acc:
0.6068 - val_loss: 0.6260 - val_acc: 0.6600
Epoch 43/100
21/21 [==============================] - 4s 211ms/step - loss: 0.7124 - acc:
0.6748 - val_loss: 0.8718 - val_acc: 0.5600
Epoch 44/100
21/21 [==============================] - 4s 200ms/step - loss: 0.7693 - acc:
```

```
0.6893 - val_loss: 0.8253 - val_acc: 0.6400
Epoch 45/100
21/21 [==============================] - 4s 207ms/step - loss: 0.6790 - acc:
0.7136 - val_loss: 0.7178 - val_acc: 0.6000
Epoch 46/100
21/21 [==============================] - 4s 209ms/step - loss: 0.6213 - acc:
0.7427 - val_loss: 0.6246 - val_acc: 0.6200
Epoch 47/100
21/21 [==============================] - 4s 210ms/step - loss: 0.7107 - acc:
0.6699 - val_loss: 0.6389 - val_acc: 0.7000
Epoch 48/100
21/21 [==============================] - 4s 212ms/step - loss: 0.6362 - acc:
0.7330 - val_loss: 0.8197 - val_acc: 0.6600
Epoch 49/100
21/21 [==============================] - 4s 207ms/step - loss: 0.7284 - acc:
0.6602 - val_loss: 0.7171 - val_acc: 0.6000
Epoch 50/100
21/21 [==============================] - 4s 198ms/step - loss: 0.6473 - acc:
0.7379 - val_loss: 0.6782 - val_acc: 0.6800
Epoch 51/100
21/21 [==============================] - 4s 206ms/step - loss: 0.6613 - acc:
0.7330 - val_loss: 0.7654 - val_acc: 0.5800
Epoch 52/100
21/21 [==============================] - 4s 207ms/step - loss: 0.6869 - acc:
0.6699 - val_loss: 0.8129 - val_acc: 0.6200
Epoch 53/100
21/21 [==============================] - 4s 209ms/step - loss: 0.6335 - acc:
0.7233 - val_loss: 0.6932 - val_acc: 0.6200
Epoch 54/100
21/21 [==============================] - 4s 209ms/step - loss: 0.6046 - acc:
0.7379 - val_loss: 0.5509 - val_acc: 0.7400
Epoch 55/100
21/21 [==============================] - 4s 213ms/step - loss: 0.6087 - acc:
0.7476 - val_loss: 0.5857 - val_acc: 0.7600
Epoch 56/100
21/21 [==============================] - 4s 205ms/step - loss: 0.6975 - acc:
0.6942 - val_loss: 0.6024 - val_acc: 0.6200
Epoch 57/100
21/21 [==============================] - 4s 207ms/step - loss: 0.6621 - acc:
0.7087 - val_loss: 0.6066 - val_acc: 0.7400
Epoch 58/100
21/21 [==============================] - 4s 207ms/step - loss: 0.7088 - acc:
0.6699 - val_loss: 0.6178 - val_acc: 0.7000
Epoch 59/100
21/21 [==============================] - 4s 214ms/step - loss: 0.6877 - acc:
0.7136 - val_loss: 0.7264 - val_acc: 0.5800
Epoch 60/100
21/21 [==============================] - 4s 211ms/step - loss: 0.6865 - acc:
```

```
0.6408 - val_loss: 0.6869 - val_acc: 0.6800
Epoch 61/100
21/21 [==============================] - 4s 210ms/step - loss: 0.6442 - acc:
0.7233 - val_loss: 0.6402 - val_acc: 0.6800
Epoch 62/100
21/21 [==============================] - 4s 203ms/step - loss: 0.6544 - acc:
0.7233 - val_loss: 0.6267 - val_acc: 0.6800
Epoch 63/100
21/21 [==============================] - 4s 208ms/step - loss: 0.6709 - acc:
0.7039 - val_loss: 0.6741 - val_acc: 0.7000
Epoch 64/100
21/21 [==============================] - 4s 205ms/step - loss: 0.6935 - acc:
0.6942 - val_loss: 0.5262 - val_acc: 0.7200
Epoch 65/100
21/21 [==============================] - 4s 203ms/step - loss: 0.6257 - acc:
0.7330 - val_loss: 0.5790 - val_acc: 0.6600
Epoch 66/100
21/21 [==============================] - 4s 201ms/step - loss: 0.6337 - acc:
0.7427 - val_loss: 0.6562 - val_acc: 0.6800
Epoch 67/100
21/21 [==============================] - 4s 206ms/step - loss: 0.6702 - acc:
0.6990 - val_loss: 0.6637 - val_acc: 0.6200
Epoch 68/100
21/21 [==============================] - 4s 210ms/step - loss: 0.6404 - acc:
0.7379 - val_loss: 0.7347 - val_acc: 0.6200
Epoch 69/100
21/21 [==============================] - 4s 196ms/step - loss: 0.6357 - acc:
0.7330 - val_loss: 0.6709 - val_acc: 0.6200
Epoch 70/100
21/21 [==============================] - 4s 203ms/step - loss: 0.6582 - acc:
0.7087 - val_loss: 0.6787 - val_acc: 0.6200
Epoch 71/100
21/21 [==============================] - 4s 209ms/step - loss: 0.6205 - acc:
0.7143 - val_loss: 0.6814 - val_acc: 0.6200
Epoch 72/100
21/21 [==============================] - 4s 211ms/step - loss: 0.6946 - acc:
0.7233 - val_loss: 0.6814 - val_acc: 0.6400
Epoch 73/100
21/21 [==============================] - 4s 208ms/step - loss: 0.6630 - acc:
0.7330 - val_loss: 0.7075 - val_acc: 0.6200
Epoch 74/100
21/21 [==============================] - 4s 210ms/step - loss: 0.6394 - acc:
0.7190 - val_loss: 0.7329 - val_acc: 0.5800
Epoch 75/100
21/21 [==============================] - 4s 203ms/step - loss: 0.6399 - acc:
0.7524 - val_loss: 0.7497 - val_acc: 0.6000
Epoch 76/100
21/21 [==============================] - 4s 206ms/step - loss: 0.6041 - acc:
```

```
0.7087 - val_loss: 0.6291 - val_acc: 0.6600
Epoch 77/100
21/21 [==============================] - 4s 205ms/step - loss: 0.6114 - acc:
0.7524 - val_loss: 0.6716 - val_acc: 0.6600
Epoch 78/100
21/21 [==============================] - 4s 204ms/step - loss: 0.6567 - acc:
0.6796 - val_loss: 0.6046 - val_acc: 0.6800
Epoch 79/100
21/21 [==============================] - 4s 203ms/step - loss: 0.6920 - acc:
0.6748 - val_loss: 0.6569 - val_acc: 0.7600
Epoch 80/100
21/21 [==============================] - 4s 206ms/step - loss: 0.6173 - acc:
0.7573 - val_loss: 0.6738 - val_acc: 0.6600
Epoch 81/100
21/21 [==============================] - 4s 202ms/step - loss: 0.5361 - acc:
0.7816 - val_loss: 0.7176 - val_acc: 0.6000
Epoch 82/100
21/21 [==============================] - 4s 206ms/step - loss: 0.6151 - acc:
0.7767 - val_loss: 0.6814 - val_acc: 0.6000
Epoch 83/100
21/21 [==============================] - 4s 201ms/step - loss: 0.6492 - acc:
0.6845 - val_loss: 0.6458 - val_acc: 0.6400
Epoch 84/100
21/21 [==============================] - 4s 208ms/step - loss: 0.6183 - acc:
0.7282 - val_loss: 0.5875 - val_acc: 0.6400
Epoch 85/100
21/21 [==============================] - 4s 208ms/step - loss: 0.5455 - acc:
0.7524 - val_loss: 0.6168 - val_acc: 0.7000
Epoch 86/100
21/21 [==============================] - 4s 205ms/step - loss: 0.6215 - acc:
0.7524 - val_loss: 0.6491 - val_acc: 0.7200
Epoch 87/100
21/21 [==============================] - 5s 217ms/step - loss: 0.5909 - acc:
0.7381 - val_loss: 0.5928 - val_acc: 0.7200
Epoch 88/100
21/21 [==============================] - 4s 202ms/step - loss: 0.6770 - acc:
0.7282 - val_loss: 0.6466 - val_acc: 0.7000
Epoch 89/100
21/21 [==============================] - 4s 213ms/step - loss: 0.6102 - acc:
0.7333 - val_loss: 0.7878 - val_acc: 0.5600
Epoch 90/100
21/21 [==============================] - 4s 213ms/step - loss: 0.6031 - acc:
0.7330 - val_loss: 0.7039 - val_acc: 0.5600
Epoch 91/100
21/21 [==============================] - 4s 210ms/step - loss: 0.6206 - acc:
0.7039 - val_loss: 0.5920 - val_acc: 0.6600
Epoch 92/100
21/21 [==============================] - 4s 210ms/step - loss: 0.6619 - acc:
```

```
0.6990 - val_loss: 0.6930 - val_acc: 0.7000
Epoch 93/100
21/21 [==============================] - 4s 210ms/step - loss: 0.5831 - acc:
0.7718 - val_loss: 0.6777 - val_acc: 0.6200
Epoch 94/100
21/21 [==============================] - 5s 214ms/step - loss: 0.6552 - acc:
0.7095 - val_loss: 0.7944 - val_acc: 0.6200
Epoch 95/100
21/21 [==============================] - 4s 208ms/step - loss: 0.6059 - acc:
0.7136 - val_loss: 0.6348 - val_acc: 0.7000
Epoch 96/100
21/21 [==============================] - 4s 210ms/step - loss: 0.5533 - acc:
0.7379 - val_loss: 0.6246 - val_acc: 0.6400
Epoch 97/100
21/21 [==============================] - 4s 201ms/step - loss: 0.5948 - acc:
0.7330 - val_loss: 0.6681 - val_acc: 0.6600
Epoch 98/100
21/21 [==============================] - 4s 198ms/step - loss: 0.6907 - acc:
0.6893 - val_loss: 0.6761 - val_acc: 0.5800
Epoch 99/100
21/21 [==============================] - 4s 206ms/step - loss: 0.5685 - acc:
0.7621 - val_loss: 0.5873 - val_acc: 0.6800
Epoch 100/100
21/21 [==============================] - 4s 203ms/step - loss: 0.5913 - acc:
0.7379 - val_loss: 0.6987 - val_acc: 0.6600
Epoch 1/100
21/21 [==============================] - 5s 237ms/step - loss: 1.7365 - acc:
0.4660 - val_loss: 1.1046 - val_acc: 0.6000
Epoch 2/100
21/21 [==============================] - 4s 204ms/step - loss: 0.9878 - acc:
0.6602 - val_loss: 1.0523 - val_acc: 0.5200
Epoch 3/100
21/21 [==============================] - 4s 201ms/step - loss: 1.0684 - acc:
0.6286 - val_loss: 0.8653 - val_acc: 0.6800
Epoch 4/100
21/21 [==============================] - 4s 208ms/step - loss: 0.8306 - acc:
0.6796 - val_loss: 0.9470 - val_acc: 0.5800
Epoch 5/100
21/21 [==============================] - 4s 205ms/step - loss: 0.8080 - acc:
0.6602 - val_loss: 1.1078 - val_acc: 0.5800
Epoch 6/100
21/21 [==============================] - 4s 210ms/step - loss: 0.8583 - acc:
0.6190 - val_loss: 0.5672 - val_acc: 0.7800
Epoch 7/100
21/21 [==============================] - 4s 210ms/step - loss: 0.8393 - acc:
0.6311 - val_loss: 0.6254 - val_acc: 0.7000
Epoch 8/100
21/21 [==============================] - 4s 205ms/step - loss: 0.7930 - acc:
```

```
0.6117 - val_loss: 0.8282 - val_acc: 0.6200
Epoch 9/100
21/21 [==============================] - 4s 199ms/step - loss: 0.6196 - acc:
0.7233 - val_loss: 0.6507 - val_acc: 0.7600
Epoch 10/100
21/21 [==============================] - 4s 210ms/step - loss: 0.5738 - acc:
0.7670 - val_loss: 0.6301 - val_acc: 0.7400
Epoch 11/100
21/21 [==============================] - 4s 203ms/step - loss: 0.5945 - acc:
0.7136 - val_loss: 0.6299 - val_acc: 0.7000
Epoch 12/100
21/21 [==============================] - 4s 204ms/step - loss: 0.6285 - acc:
0.7718 - val_loss: 0.6369 - val_acc: 0.7400
Epoch 13/100
21/21 [==============================] - 4s 203ms/step - loss: 0.5935 - acc:
0.7621 - val_loss: 0.7478 - val_acc: 0.6600
Epoch 14/100
21/21 [==============================] - 4s 209ms/step - loss: 0.6000 - acc:
0.7233 - val_loss: 0.8489 - val_acc: 0.5800
Epoch 15/100
21/21 [==============================] - 4s 208ms/step - loss: 0.6348 - acc:
0.7330 - val_loss: 0.7561 - val_acc: 0.7600
Epoch 16/100
21/21 [==============================] - 4s 205ms/step - loss: 0.5507 - acc:
0.7184 - val_loss: 0.5462 - val_acc: 0.7000
Epoch 17/100
21/21 [==============================] - 4s 207ms/step - loss: 0.5932 - acc:
0.7476 - val_loss: 0.6922 - val_acc: 0.7600
Epoch 18/100
21/21 [==============================] - 4s 209ms/step - loss: 0.5579 - acc:
0.7913 - val_loss: 0.6477 - val_acc: 0.8000
Epoch 19/100
21/21 [==============================] - 4s 202ms/step - loss: 0.5666 - acc:
0.7573 - val_loss: 0.8428 - val_acc: 0.6000
Epoch 20/100
21/21 [==============================] - 4s 208ms/step - loss: 0.5965 - acc:
0.7621 - val_loss: 0.6930 - val_acc: 0.6600
Epoch 21/100
21/21 [==============================] - 4s 205ms/step - loss: 0.5484 - acc:
0.7379 - val_loss: 0.8201 - val_acc: 0.7200
Epoch 22/100
21/21 [==============================] - 4s 206ms/step - loss: 0.5950 - acc:
0.7718 - val_loss: 0.7863 - val_acc: 0.6000
Epoch 23/100
21/21 [==============================] - 4s 207ms/step - loss: 0.6375 - acc:
0.7136 - val_loss: 0.9563 - val_acc: 0.6400
Epoch 24/100
21/21 [==============================] - 4s 207ms/step - loss: 0.5809 - acc:
```

```
0.7524 - val_loss: 0.7653 - val_acc: 0.7000
Epoch 25/100
21/21 [==============================] - 4s 208ms/step - loss: 0.5539 - acc:
0.7718 - val_loss: 0.5677 - val_acc: 0.7600
Epoch 26/100
21/21 [==============================] - 4s 203ms/step - loss: 0.5378 - acc:
0.7621 - val_loss: 0.8747 - val_acc: 0.6600
Epoch 27/100
21/21 [==============================] - 4s 207ms/step - loss: 0.5143 - acc:
0.7573 - val_loss: 0.7813 - val_acc: 0.6400
Epoch 28/100
21/21 [==============================] - 4s 204ms/step - loss: 0.5125 - acc:
0.8010 - val_loss: 0.8947 - val_acc: 0.6000
Epoch 29/100
21/21 [==============================] - 4s 200ms/step - loss: 0.5218 - acc:
0.7913 - val_loss: 0.6905 - val_acc: 0.6800
Epoch 30/100
21/21 [==============================] - 4s 207ms/step - loss: 0.4459 - acc:
0.7767 - val_loss: 0.8545 - val_acc: 0.6400
Epoch 31/100
21/21 [==============================] - 4s 205ms/step - loss: 0.5355 - acc:
0.7767 - val_loss: 0.9406 - val_acc: 0.7200
Epoch 32/100
21/21 [==============================] - 4s 209ms/step - loss: 0.5553 - acc:
0.7718 - val_loss: 0.7004 - val_acc: 0.6400
Epoch 33/100
21/21 [==============================] - 4s 207ms/step - loss: 0.5082 - acc:
0.7767 - val_loss: 0.9647 - val_acc: 0.6200
Epoch 34/100
21/21 [==============================] - 4s 203ms/step - loss: 0.4922 - acc:
0.7718 - val_loss: 0.7913 - val_acc: 0.6600
Epoch 35/100
21/21 [==============================] - 4s 200ms/step - loss: 0.5505 - acc:
0.7621 - val_loss: 0.6181 - val_acc: 0.6400
Epoch 36/100
21/21 [==============================] - 4s 207ms/step - loss: 0.5558 - acc:
0.7136 - val_loss: 0.5996 - val_acc: 0.7000
Epoch 37/100
21/21 [==============================] - 4s 210ms/step - loss: 0.4712 - acc:
0.8010 - val_loss: 0.6383 - val_acc: 0.7200
Epoch 38/100
21/21 [==============================] - 4s 209ms/step - loss: 0.4499 - acc:
0.8204 - val_loss: 0.6481 - val_acc: 0.8400
Epoch 39/100
21/21 [==============================] - 4s 204ms/step - loss: 0.4768 - acc:
0.8010 - val_loss: 0.7860 - val_acc: 0.6800
Epoch 40/100
21/21 [==============================] - 4s 200ms/step - loss: 0.4852 - acc:
```

```
0.7961 - val_loss: 0.6336 - val_acc: 0.6400
Epoch 41/100
21/21 [==============================] - 4s 211ms/step - loss: 0.4667 - acc:
0.8000 - val_loss: 0.9014 - val_acc: 0.6800
Epoch 42/100
21/21 [==============================] - 4s 206ms/step - loss: 0.5887 - acc:
0.7816 - val_loss: 0.7073 - val_acc: 0.6200
Epoch 43/100
21/21 [==============================] - 4s 203ms/step - loss: 0.4171 - acc:
0.7864 - val_loss: 0.6488 - val_acc: 0.7000
Epoch 44/100
21/21 [==============================] - 4s 205ms/step - loss: 0.4099 - acc:
0.8010 - val_loss: 0.8210 - val_acc: 0.6200
Epoch 45/100
21/21 [==============================] - 4s 206ms/step - loss: 0.4512 - acc:
0.8155 - val_loss: 0.6907 - val_acc: 0.6400
Epoch 46/100
21/21 [==============================] - 4s 198ms/step - loss: 0.4757 - acc:
0.7864 - val_loss: 0.9097 - val_acc: 0.5800
Epoch 47/100
21/21 [==============================] - 4s 212ms/step - loss: 0.4867 - acc:
0.8238 - val_loss: 0.6571 - val_acc: 0.6600
Epoch 48/100
21/21 [==============================] - 4s 213ms/step - loss: 0.4600 - acc:
0.7767 - val_loss: 0.7815 - val_acc: 0.6400
Epoch 49/100
21/21 [==============================] - 4s 202ms/step - loss: 0.5243 - acc:
0.7767 - val_loss: 0.7622 - val_acc: 0.5400
Epoch 50/100
21/21 [==============================] - 4s 198ms/step - loss: 0.4456 - acc:
0.7961 - val_loss: 0.6902 - val_acc: 0.7000
Epoch 51/100
21/21 [==============================] - 4s 195ms/step - loss: 0.4127 - acc:
0.7864 - val_loss: 0.9438 - val_acc: 0.7000
Epoch 52/100
21/21 [==============================] - 4s 210ms/step - loss: 0.4684 - acc:
0.8143 - val_loss: 0.6321 - val_acc: 0.7000
Epoch 53/100
21/21 [==============================] - 4s 206ms/step - loss: 0.5026 - acc:
0.7767 - val_loss: 0.7503 - val_acc: 0.7400
Epoch 54/100
21/21 [==============================] - 4s 207ms/step - loss: 0.4449 - acc:
0.8058 - val_loss: 0.7451 - val_acc: 0.6800
Epoch 55/100
21/21 [==============================] - 4s 202ms/step - loss: 0.4360 - acc:
0.8107 - val_loss: 0.8143 - val_acc: 0.7200
Epoch 56/100
21/21 [==============================] - 4s 198ms/step - loss: 0.4965 - acc:
```

```
0.7816 - val_loss: 0.5605 - val_acc: 0.7600
Epoch 57/100
21/21 [==============================] - 4s 204ms/step - loss: 0.4765 - acc:
0.8155 - val_loss: 0.8237 - val_acc: 0.6400
Epoch 58/100
21/21 [==============================] - 4s 212ms/step - loss: 0.4060 - acc:
0.8190 - val_loss: 0.7667 - val_acc: 0.7000
Epoch 59/100
21/21 [==============================] - 4s 207ms/step - loss: 0.4629 - acc:
0.8010 - val_loss: 0.5993 - val_acc: 0.7200
Epoch 60/100
21/21 [==============================] - 4s 205ms/step - loss: 0.4439 - acc:
0.8155 - val_loss: 0.6009 - val_acc: 0.7800
Epoch 61/100
21/21 [==============================] - 4s 205ms/step - loss: 0.4770 - acc:
0.7961 - val_loss: 0.6179 - val_acc: 0.7400
Epoch 62/100
21/21 [==============================] - 4s 201ms/step - loss: 0.4541 - acc:
0.8058 - val_loss: 0.6877 - val_acc: 0.6800
Epoch 63/100
21/21 [==============================] - 4s 201ms/step - loss: 0.3586 - acc:
0.8592 - val_loss: 0.7067 - val_acc: 0.6800
Epoch 64/100
21/21 [==============================] - 4s 206ms/step - loss: 0.4831 - acc:
0.8155 - val_loss: 0.6713 - val_acc: 0.7000
Epoch 65/100
21/21 [==============================] - 4s 209ms/step - loss: 0.4290 - acc:
0.8204 - val_loss: 0.7578 - val_acc: 0.6400
Epoch 66/100
21/21 [==============================] - 4s 211ms/step - loss: 0.5062 - acc:
0.7913 - val_loss: 0.7209 - val_acc: 0.6400
Epoch 67/100
21/21 [==============================] - 4s 208ms/step - loss: 0.4285 - acc:
0.8058 - val_loss: 0.6567 - val_acc: 0.7400
Epoch 68/100
21/21 [==============================] - 4s 200ms/step - loss: 0.4715 - acc:
0.8204 - val_loss: 0.7281 - val_acc: 0.7000
Epoch 69/100
21/21 [==============================] - 4s 208ms/step - loss: 0.4119 - acc:
0.8447 - val_loss: 0.6881 - val_acc: 0.6600
Epoch 70/100
21/21 [==============================] - 4s 211ms/step - loss: 0.3258 - acc:
0.8786 - val_loss: 0.8021 - val_acc: 0.7000
Epoch 71/100
21/21 [==============================] - 4s 209ms/step - loss: 0.4484 - acc:
0.7864 - val_loss: 0.5994 - val_acc: 0.6600
Epoch 72/100
21/21 [==============================] - 4s 208ms/step - loss: 0.4165 - acc:
```

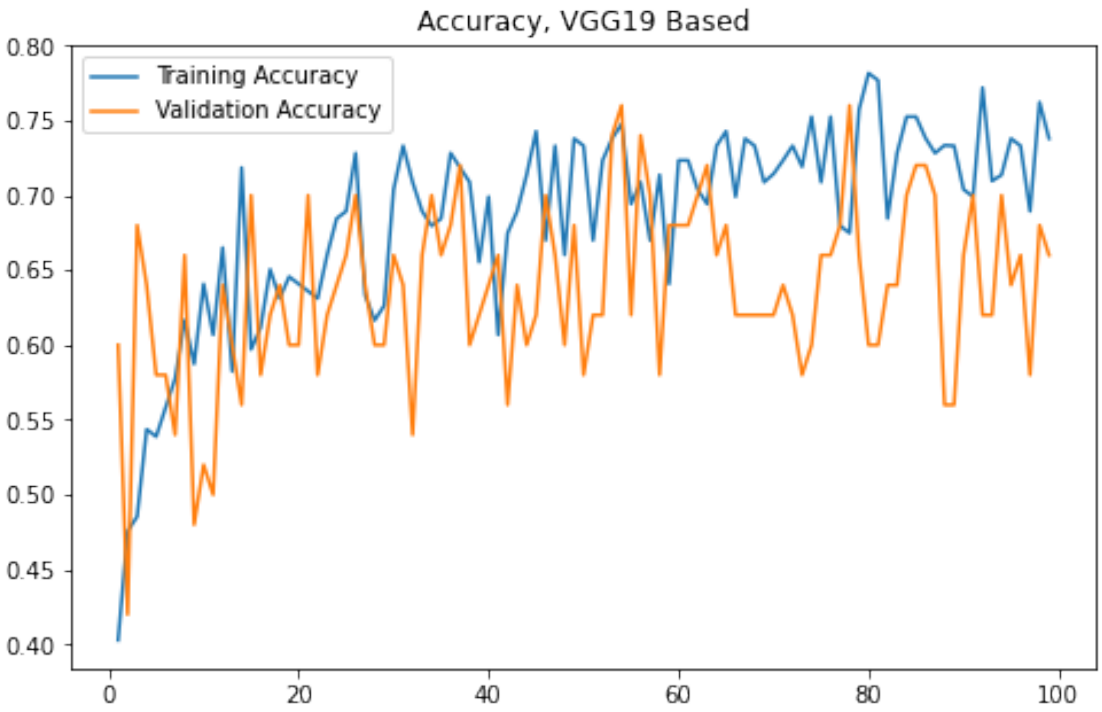
```
0.8301 - val_loss: 0.6224 - val_acc: 0.7000
Epoch 73/100
21/21 [==============================] - 4s 207ms/step - loss: 0.4178 - acc:
0.8592 - val_loss: 0.6952 - val_acc: 0.7000
Epoch 74/100
21/21 [==============================] - 4s 205ms/step - loss: 0.4223 - acc:
0.8301 - val_loss: 0.8498 - val_acc: 0.6200
Epoch 75/100
21/21 [==============================] - 4s 202ms/step - loss: 0.4034 - acc:
0.8544 - val_loss: 0.7555 - val_acc: 0.7200
Epoch 76/100
21/21 [==============================] - 4s 211ms/step - loss: 0.5054 - acc:
0.7913 - val_loss: 0.7318 - val_acc: 0.6200
Epoch 77/100
21/21 [==============================] - 4s 203ms/step - loss: 0.4772 - acc:
0.7913 - val_loss: 0.5014 - val_acc: 0.7000
Epoch 78/100
21/21 [==============================] - 4s 209ms/step - loss: 0.3890 - acc:
0.8398 - val_loss: 0.6194 - val_acc: 0.7200
Epoch 79/100
21/21 [==============================] - 4s 208ms/step - loss: 0.4501 - acc:
0.8058 - val_loss: 0.5586 - val_acc: 0.7200
Epoch 80/100
21/21 [==============================] - 4s 208ms/step - loss: 0.4100 - acc:
0.8252 - val_loss: 0.6990 - val_acc: 0.7000
Epoch 81/100
21/21 [==============================] - 4s 209ms/step - loss: 0.3643 - acc:
0.8447 - val_loss: 0.8733 - val_acc: 0.6000
Epoch 82/100
21/21 [==============================] - 4s 206ms/step - loss: 0.3911 - acc:
0.8447 - val_loss: 0.5383 - val_acc: 0.7400
Epoch 83/100
21/21 [==============================] - 4s 205ms/step - loss: 0.4559 - acc:
0.8058 - val_loss: 0.5837 - val_acc: 0.7200
Epoch 84/100
21/21 [==============================] - 4s 211ms/step - loss: 0.3869 - acc:
0.8204 - val_loss: 0.7078 - val_acc: 0.7200
Epoch 85/100
21/21 [==============================] - 4s 203ms/step - loss: 0.3817 - acc:
0.8058 - val_loss: 0.6806 - val_acc: 0.7200
Epoch 86/100
21/21 [==============================] - 4s 201ms/step - loss: 0.3828 - acc:
0.8301 - val_loss: 0.7237 - val_acc: 0.6800
Epoch 87/100
21/21 [==============================] - 4s 206ms/step - loss: 0.4122 - acc:
0.8301 - val_loss: 0.6673 - val_acc: 0.7200
Epoch 88/100
21/21 [==============================] - 4s 207ms/step - loss: 0.3817 - acc:
```

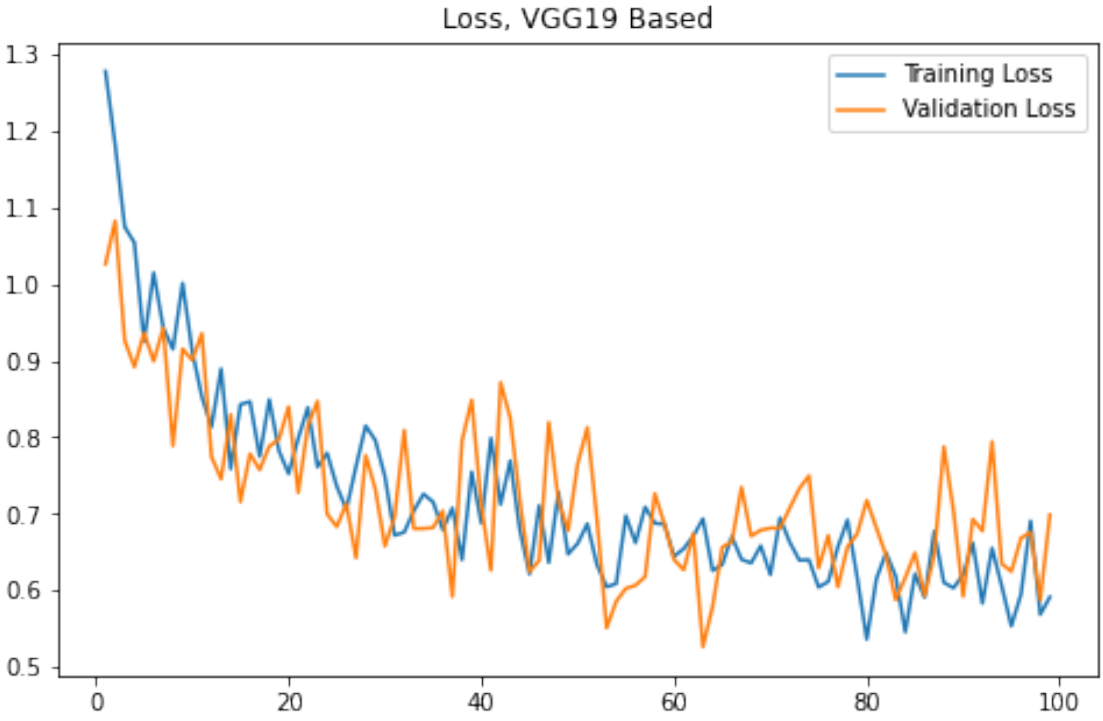
```
0.8398 - val_loss: 0.4581 - val_acc: 0.8400
Epoch 89/100
21/21 [==============================] - 4s 210ms/step - loss: 0.3886 - acc:
0.8155 - val_loss: 0.6943 - val_acc: 0.7000
Epoch 90/100
21/21 [==============================] - 4s 205ms/step - loss: 0.3823 - acc:
0.8204 - val_loss: 0.6030 - val_acc: 0.7600
Epoch 91/100
21/21 [==============================] - 4s 205ms/step - loss: 0.4127 - acc:
0.8301 - val_loss: 0.8126 - val_acc: 0.6800
Epoch 92/100
21/21 [==============================] - 4s 206ms/step - loss: 0.3820 - acc:
0.8155 - val_loss: 0.9313 - val_acc: 0.6000
Epoch 93/100
21/21 [==============================] - 4s 211ms/step - loss: 0.4534 - acc:
0.7961 - val_loss: 0.6263 - val_acc: 0.7200
Epoch 94/100
21/21 [==============================] - 4s 210ms/step - loss: 0.3687 - acc:
0.8544 - val_loss: 0.4769 - val_acc: 0.8200
Epoch 95/100
21/21 [==============================] - 4s 208ms/step - loss: 0.3292 - acc:
0.8592 - val_loss: 0.9060 - val_acc: 0.6400
Epoch 96/100
21/21 [==============================] - 4s 208ms/step - loss: 0.4616 - acc:
0.8107 - val_loss: 0.8084 - val_acc: 0.6800
Epoch 97/100
21/21 [==============================] - 4s 209ms/step - loss: 0.4448 - acc:
0.8010 - val_loss: 0.7275 - val_acc: 0.6800
Epoch 98/100
21/21 [==============================] - 4s 205ms/step - loss: 0.4353 - acc:
0.7961 - val_loss: 0.7070 - val_acc: 0.7200
Epoch 99/100
21/21 [==============================] - 4s 213ms/step - loss: 0.4269 - acc:
0.8495 - val_loss: 0.7074 - val_acc: 0.7600
Epoch 100/100
21/21 [==============================] - 4s 206ms/step - loss: 0.3992 - acc:
0.8010 - val_loss: 0.6009 - val_acc: 0.7200
```

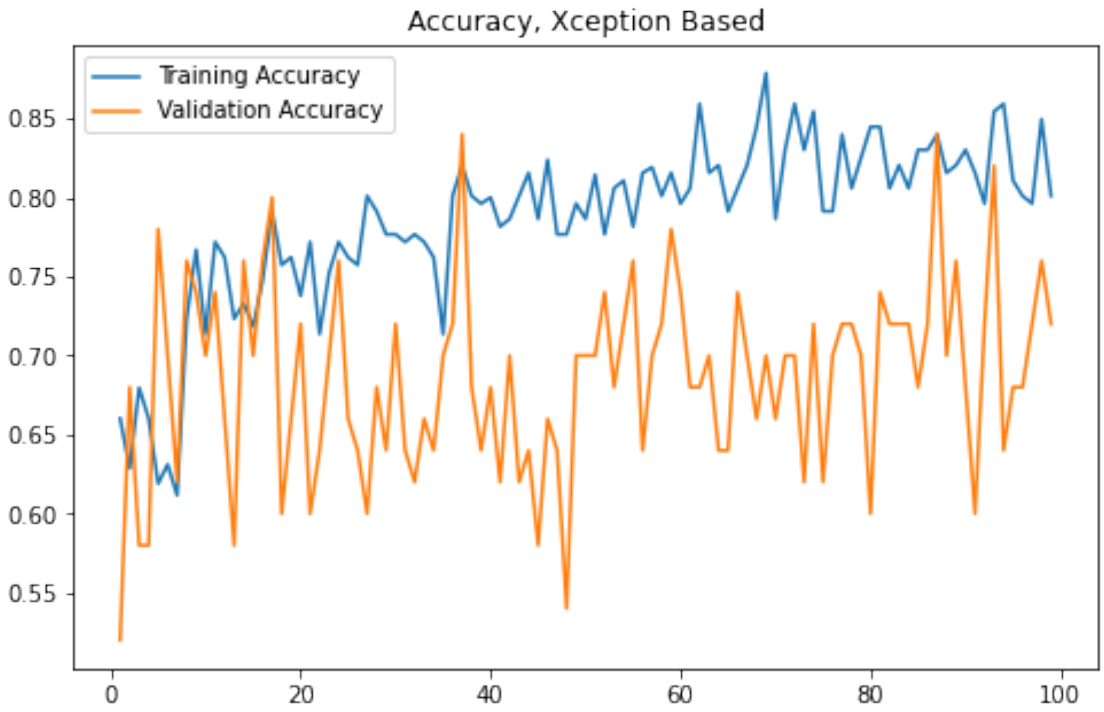
[5 points] Plot Accuracy and Loss During Training

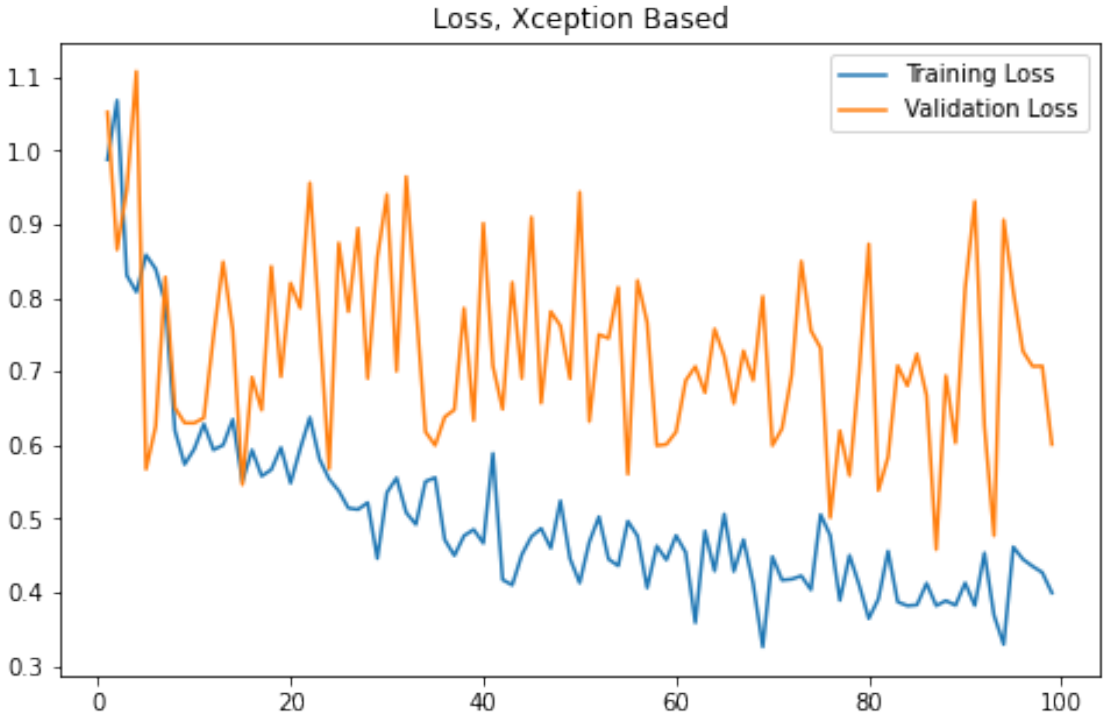
[26]:
```
import matplotlib.pyplot as plt

acc = fitted.history['acc']
loss = fitted.history['loss']
val_acc = fitted.history['val_acc']
val_loss = fitted.history['val_loss']

# accuracy plot
```

```
plt.figure(figsize=(8, 5))
plt.plot(np.arange(1,NUM_EPOCHS), acc[1:], label='Training Accuracy')
plt.plot(np.arange(1,NUM_EPOCHS), val_acc[1:], label='Validation Accuracy')
plt.title('Accuracy, VGG19 Based')
plt.legend()
plt.show()

# loss plot
plt.figure(figsize=(8, 5))
plt.plot(np.arange(1,NUM_EPOCHS), loss[1:], label='Training Loss')
plt.plot(np.arange(1,NUM_EPOCHS), val_loss[1:], label='Validation Loss')
plt.title('Loss, VGG19 Based')
plt.legend()
plt.show()



acc2 = fitted2.history['acc']
loss2 = fitted2.history['loss']
val_acc2 = fitted2.history['val_acc']
val_loss2 = fitted2.history['val_loss']

# accuracy plot
plt.figure(figsize=(8, 5))
plt.plot(np.arange(1,NUM_EPOCHS), acc2[1:], label='Training Accuracy')
plt.plot(np.arange(1,NUM_EPOCHS), val_acc2[1:], label='Validation Accuracy')
plt.title('Accuracy, Xception Based')
plt.legend()
plt.show()

# loss plot
plt.figure(figsize=(8, 5))
plt.plot(np.arange(1,NUM_EPOCHS), loss2[1:], label='Training Loss')
plt.plot(np.arange(1,NUM_EPOCHS), val_loss2[1:], label='Validation Loss')
plt.title('Loss, Xception Based')
plt.legend()
plt.show()
```


Accuracy, VGG19 Based
Training Accuracy
Validation Accuracy
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
20
40
60
80
100


Loss, VGG19 Based
Training Loss
Validation Loss
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100


Accuracy, Xception Based
Training Accuracy
Validation Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
20
40
60
80
100


Loss, Xception Based
Training Loss
Validation Loss
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Testing Model

[27]:
```
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.
 ↪flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                    ␣
 ↪batch_size=1,shuffle=True,seed=42,class_mode="categorical")
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 for VGG16:' , x[0])
print('Test accuracy for VGG16:',x[1])

# for ResNet Model
eval_generator.reset()
print(len(eval_generator))
x2 = model2.evaluate_generator(eval_generator,steps = np.
 ↪ceil(len(eval_generator)),
                             use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss for Xception:' , x2[0])
print('Test accuracy for Xception:',x2[1])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 1s 14ms/step - loss: 0.8207 - acc:
0.5833
Test loss for VGG16: 0.8206884264945984
Test accuracy for VGG16: 0.5833333134651184
36
36/36 [==============================] - 0s 13ms/step - loss: 0.4808 - acc:
0.7778
Test loss for ResNet: 0.48077499866485596
Test accuracy for ResNet: 0.7777777910232544
```

2.4 [10 points] TSNE Plot

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely used technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. After training is complete, extract features from a specific deep layer of your choice, use t-SNE to reduce the dimensionality of your extracted features to 2 dimensions and plot the resulting 2D features.

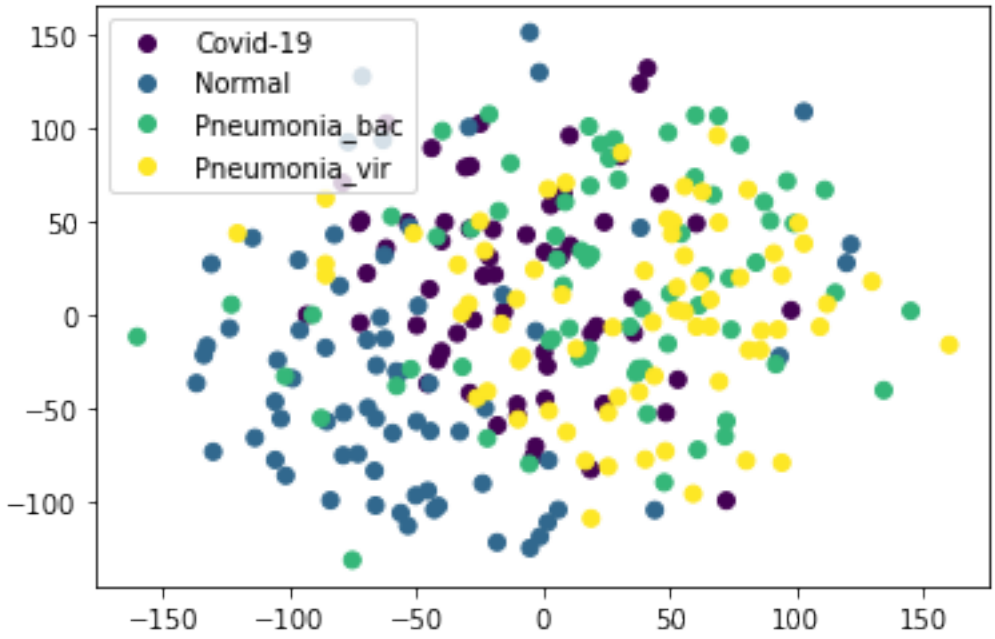
[42]:
```
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.Model(inputs=model.input,
                                          outputs=model.get_layer('dense_feature').
 ↪output)
```

```
tsne_eval_generator = test_datagen.
 ↪flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                          ␣
 ↪batch_size=1,shuffle=False,seed=42,class_mode="categorical")

y_true = tsne_eval_generator.classes

# Compile
intermediate_layer_model.compile(loss='categorical_crossentropy',␣
 ↪optimizer=keras.optimizers.Adam(lr=LEARNING_RATE), metrics=['acc'])

features = intermediate_layer_model.predict(tsne_eval_generator)

tsne = TSNE(n_components=4, method='exact', n_jobs=-1, perplexity=30,␣
 ↪learning_rate=175, n_iter=3500).fit_transform(features)

my_label = ['Covid-19', 'Normal', 'Pneumonia_bac', 'Pneumonia_vir']

scatter = plt.scatter(tsne[:,0], tsne[:,1], c=y_true)
handles, _ = scatter.legend_elements(prop='colors')
plt.legend(handles, my_label)
plt.show()
```

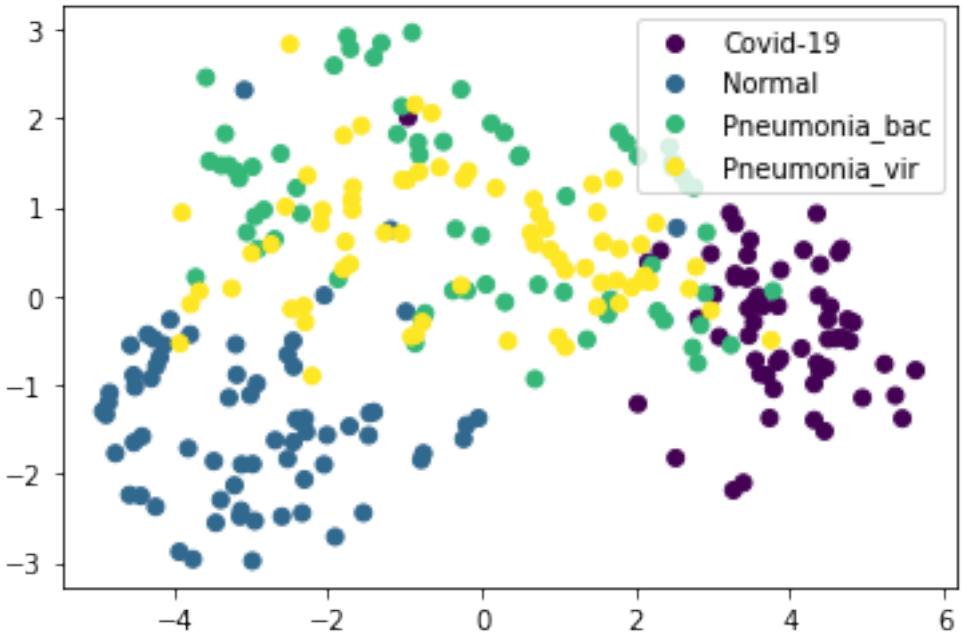
```
# For Xception
intermediate_layer_model2 = tf.keras.Model(inputs=model2.input,
                                           outputs=model2.
 ↪get_layer('dense_feature').output)

tsne_eval_generator2 = test_datagen.
 ↪flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                          ␣
 ↪batch_size=1,shuffle=False,seed=42,class_mode="categorical")

y_true2 = tsne_eval_generator2.classes

# Compile
intermediate_layer_model2.compile(loss='categorical_crossentropy',␣
 ↪optimizer=keras.optimizers.Adam(lr=LEARNING_RATE), metrics=['acc'])
```

```
features2 = intermediate_layer_model2.predict(tsne_eval_generator2)

tsne2 = TSNE(n_components=4, method='exact', n_jobs=-1, perplexity=30,␣
 ↪learning_rate=40, n_iter=3500).fit_transform(features2)

my_label2 = ['Covid-19', 'Normal', 'Pneumonia_bac', 'Pneumonia_vir']

scatter2 = plt.scatter(tsne2[:,0], tsne2[:,1], c=y_true2)
handles2, _ = scatter.legend_elements(prop='colors')
plt.legend(handles2, my_label2)
plt.show()
```

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




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


Covid-19
Normal
Pneumonia_bac
Pneumonia_vir
3
2
1
0
−1
−2
−3
−4
−2
0
2
4
6