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
```

In [ ]:

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]: '2.7.0'

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
NUM_EPOCHS    = 40
LEARNING_RATE = 0.001
```

In [3]:
```
print(NUM_CLASSES)
```

2

Generate Training and Validation Batches

In [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.
C:\Users\kpete\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.py:342: UserWarning: This ImageDataGenerator specifies `zca_whitening` which overr
ides 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.

In [5]:
```
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation, DenseFeatures, Dropout, Conv2D, MaxPool2D, Flatten

vggmodel = tf.keras.applications.VGG16(
    weights="imagenet",
    input_shape=(224, 224, 3),
    include_top=False
)
vggmodel.trainable = False



model = Sequential(
    [
        vggmodel,
        Flatten(),

        Dense(units = 256,activation='relu', name = 'dense_feature'),

        Dense(units = 1, activation = 'sigmoid', name = 'dense_1')
    ]
)

opt = tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE)
model.compile(optimizer = opt, loss=tf.keras.losses.BinaryCrossentropy(), metrics=['accuracy'])



model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 dense_feature (Dense)       (None, 256)               6422784

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 21,137,729
Trainable params: 6,423,041
Non-trainable params: 14,714,688
_________________________________________________________________
```

[5 points] Train Model

In [10]:
```
print(len(train_batches))
print(len(valid_batches))

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

history = model.fit_generator(steps_per_epoch = STEP_SIZE_TRAIN, generator = train_batches, validation_data = valid_batches, validation_steps = STEP_SIZE_VALID,
                    epochs=45
)
```

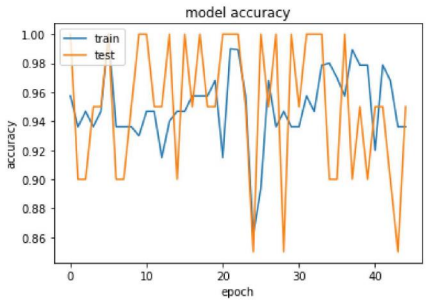
```
11
3
Epoch 1/45
C:\Users\kpete\AppData\Local\Temp/ipykernel_23760/3129912149.py:7: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(steps_per_epoch = STEP_SIZE_TRAIN, generator = train_batches, validation_data = valid_batches, validation_steps = STEP_SIZE_VALID,
10/10 [==============================] - 8s 818ms/step - loss: 0.1048 - accuracy: 0.9574 - val_loss: 0.0564 - val_accuracy: 1.0000
Epoch 2/45
10/10 [==============================] - 8s 770ms/step - loss: 0.1762 - accuracy: 0.9362 - val_loss: 0.2605 - val_accuracy: 0.9000
Epoch 3/45
10/10 [==============================] - 8s 781ms/step - loss: 0.1511 - accuracy: 0.9468 - val_loss: 0.1053 - val_accuracy: 0.9000
Epoch 4/45
10/10 [==============================] - 8s 768ms/step - loss: 0.1355 - accuracy: 0.9362 - val_loss: 0.1507 - val_accuracy: 0.9500
Epoch 5/45
10/10 [==============================] - 8s 751ms/step - loss: 0.1235 - accuracy: 0.9468 - val_loss: 0.1169 - val_accuracy: 0.9500
Epoch 6/45
10/10 [==============================] - 8s 761ms/step - loss: 0.0403 - accuracy: 1.0000 - val_loss: 0.0739 - val_accuracy: 1.0000
Epoch 7/45
10/10 [==============================] - 8s 768ms/step - loss: 0.1141 - accuracy: 0.9362 - val_loss: 0.1564 - val_accuracy: 0.9000
Epoch 8/45
10/10 [==============================] - 8s 754ms/step - loss: 0.1446 - accuracy: 0.9362 - val_loss: 0.1367 - val_accuracy: 0.9000
Epoch 9/45
10/10 [==============================] - 8s 773ms/step - loss: 0.2324 - accuracy: 0.9362 - val_loss: 0.1637 - val_accuracy: 0.9500
Epoch 10/45
10/10 [==============================] - 8s 806ms/step - loss: 0.1154 - accuracy: 0.9300 - val_loss: 0.0255 - val_accuracy: 1.0000
Epoch 11/45
10/10 [==============================] - 8s 768ms/step - loss: 0.2033 - accuracy: 0.9468 - val_loss: 0.0321 - val_accuracy: 1.0000
Epoch 12/45
10/10 [==============================] - 8s 765ms/step - loss: 0.1222 - accuracy: 0.9468 - val_loss: 0.0761 - val_accuracy: 0.9500
Epoch 13/45
10/10 [==============================] - 8s 774ms/step - loss: 0.1825 - accuracy: 0.9149 - val_loss: 0.0778 - val_accuracy: 0.9500
Epoch 14/45
10/10 [==============================] - 8s 809ms/step - loss: 0.1499 - accuracy: 0.9400 - val_loss: 0.0129 - val_accuracy: 1.0000
Epoch 15/45
10/10 [==============================] - 8s 762ms/step - loss: 0.1978 - accuracy: 0.9468 - val_loss: 0.3064 - val_accuracy: 0.9000
Epoch 16/45
10/10 [==============================] - 8s 771ms/step - loss: 0.1108 - accuracy: 0.9468 - val_loss: 0.0569 - val_accuracy: 1.0000
Epoch 17/45
10/10 [==============================] - 8s 760ms/step - loss: 0.1112 - accuracy: 0.9574 - val_loss: 0.0496 - val_accuracy: 0.9500
Epoch 18/45
10/10 [==============================] - 8s 782ms/step - loss: 0.1389 - accuracy: 0.9574 - val_loss: 0.0350 - val_accuracy: 1.0000
Epoch 19/45
```

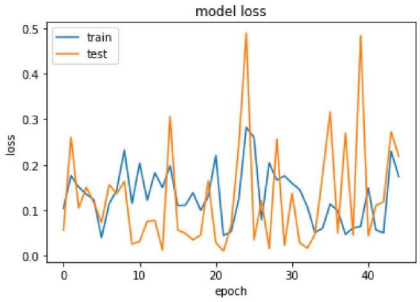
```
10/10 [==============================] - 8s 755ms/step - loss: 0.1003 - accuracy: 0.9574 - val_loss: 0.0458 - val_accuracy: 0.9500
Epoch 20/45
10/10 [==============================] - 8s 782ms/step - loss: 0.1303 - accuracy: 0.9681 - val_loss: 0.1648 - val_accuracy: 0.9500
Epoch 21/45
10/10 [==============================] - 8s 807ms/step - loss: 0.2204 - accuracy: 0.9149 - val_loss: 0.0288 - val_accuracy: 1.0000
Epoch 22/45
10/10 [==============================] - 8s 813ms/step - loss: 0.0448 - accuracy: 0.9900 - val_loss: 0.0109 - val_accuracy: 1.0000
Epoch 23/45
10/10 [==============================] - 8s 754ms/step - loss: 0.0532 - accuracy: 0.9894 - val_loss: 0.0663 - val_accuracy: 1.0000
Epoch 24/45
10/10 [==============================] - 8s 750ms/step - loss: 0.1231 - accuracy: 0.9574 - val_loss: 0.2400 - val_accuracy: 0.9500
Epoch 25/45
10/10 [==============================] - 8s 752ms/step - loss: 0.2823 - accuracy: 0.8617 - val_loss: 0.4894 - val_accuracy: 0.8500
Epoch 26/45
10/10 [==============================] - 8s 788ms/step - loss: 0.2603 - accuracy: 0.8936 - val_loss: 0.0351 - val_accuracy: 1.0000
Epoch 27/45
10/10 [==============================] - 8s 831ms/step - loss: 0.0790 - accuracy: 0.9681 - val_loss: 0.1215 - val_accuracy: 0.9500
Epoch 28/45
10/10 [==============================] - 8s 813ms/step - loss: 0.2042 - accuracy: 0.9362 - val_loss: 0.0162 - val_accuracy: 1.0000
Epoch 29/45
10/10 [==============================] - 8s 771ms/step - loss: 0.1666 - accuracy: 0.9468 - val_loss: 0.2564 - val_accuracy: 0.8500
Epoch 30/45
10/10 [==============================] - 8s 766ms/step - loss: 0.1758 - accuracy: 0.9362 - val_loss: 0.0233 - val_accuracy: 1.0000
Epoch 31/45
10/10 [==============================] - 8s 758ms/step - loss: 0.1594 - accuracy: 0.9362 - val_loss: 0.1371 - val_accuracy: 0.9500
Epoch 32/45
10/10 [==============================] - 8s 783ms/step - loss: 0.1455 - accuracy: 0.9574 - val_loss: 0.0287 - val_accuracy: 1.0000
Epoch 33/45
10/10 [==============================] - 8s 829ms/step - loss: 0.1083 - accuracy: 0.9468 - val_loss: 0.0172 - val_accuracy: 1.0000
Epoch 34/45
10/10 [==============================] - 8s 826ms/step - loss: 0.0516 - accuracy: 0.9787 - val_loss: 0.0470 - val_accuracy: 1.0000
Epoch 35/45
10/10 [==============================] - 8s 830ms/step - loss: 0.0605 - accuracy: 0.9800 - val_loss: 0.1770 - val_accuracy: 0.9000
Epoch 36/45
10/10 [==============================] - 8s 815ms/step - loss: 0.1138 - accuracy: 0.9700 - val_loss: 0.3165 - val_accuracy: 0.9000
Epoch 37/45
10/10 [==============================] - 8s 773ms/step - loss: 0.0994 - accuracy: 0.9574 - val_loss: 0.0501 - val_accuracy: 1.0000
Epoch 38/45
10/10 [==============================] - 8s 779ms/step - loss: 0.0473 - accuracy: 0.9894 - val_loss: 0.2695 - val_accuracy: 0.9000
Epoch 39/45
10/10 [==============================] - 8s 789ms/step - loss: 0.0612 - accuracy: 0.9787 - val_loss: 0.0461 - val_accuracy: 0.9500
Epoch 40/45
10/10 [==============================] - 8s 770ms/step - loss: 0.0648 - accuracy: 0.9787 - val_loss: 0.4832 - val_accuracy: 0.9000
Epoch 41/45
10/10 [==============================] - 8s 790ms/step - loss: 0.1491 - accuracy: 0.9200 - val_loss: 0.0440 - val_accuracy: 0.9500
Epoch 42/45
10/10 [==============================] - 8s 771ms/step - loss: 0.0569 - accuracy: 0.9787 - val_loss: 0.1107 - val_accuracy: 0.9500
Epoch 43/45
10/10 [==============================] - 8s 762ms/step - loss: 0.0505 - accuracy: 0.9681 - val_loss: 0.1187 - val_accuracy: 0.9000
Epoch 44/45
10/10 [==============================] - 8s 808ms/step - loss: 0.2296 - accuracy: 0.9362 - val_loss: 0.2724 - val_accuracy: 0.8500
Epoch 45/45
10/10 [==============================] - 8s 776ms/step - loss: 0.1740 - accuracy: 0.9362 - val_loss: 0.2189 - val_accuracy: 0.9500
```

[5 points] Plot Accuracy and Loss During Training

In [13]:

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()


plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



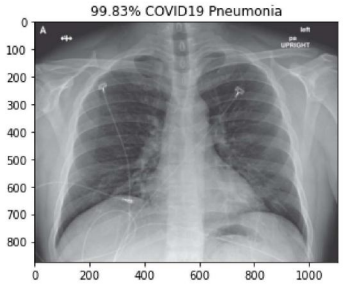


In [ ]:

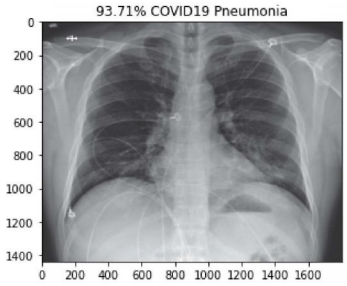
Plot Test Results

In [14]:
```
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()
```

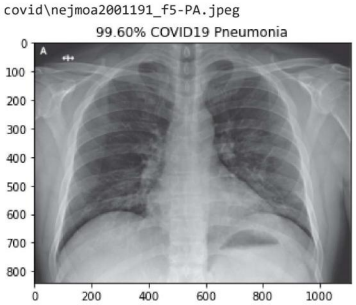
```
Found 18 images belonging to 2 classes.
 1/18 [>.............................] - ETA: 1s
C:\Users\kpete\AppData\Local\Temp/ipykernel_23760/1882219165.py:7: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future version. Please use
`Model.predict`, which supports generators.
  pred = model.predict_generator(eval_generator,18,verbose=1)
18/18 [==============================] - 2s 82ms/step
covid\nejmoa2001191_f3-PA.jpeg
```



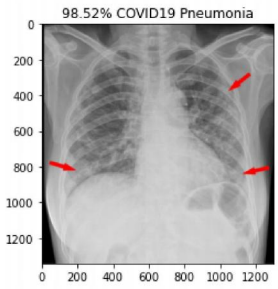

```
covid\nejmoa2001191_f4.jpeg
```



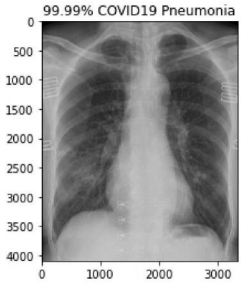

```
covid\nejmoa2001191_f5-PA.jpeg
```



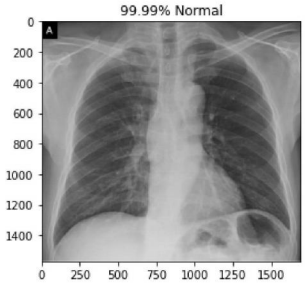

```
covid\radiol.2020200490.fig3.jpeg
```



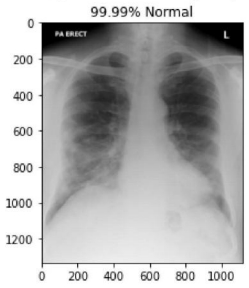
covid\ryct.2020200028.fig1a.jpeg



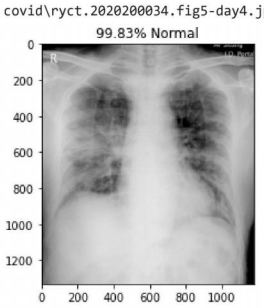
covid\ryct.2020200034.fig2.jpeg



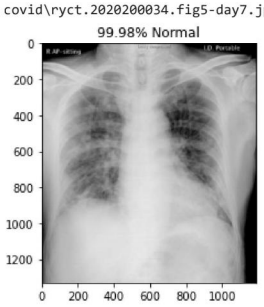
covid\ryct.2020200034.fig5-day0.jpeg



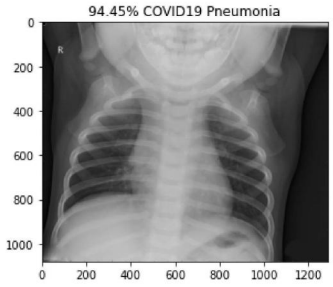
covid\ryct.2020200034.fig5-day4.jpeg



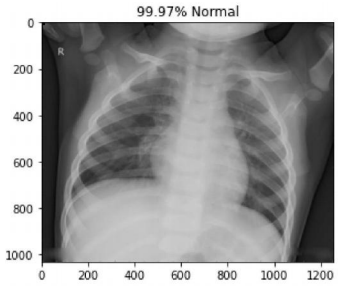
covid\ryct.2020200034.fig5-day7.jpeg



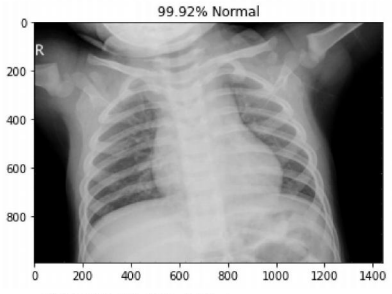
normal\NORMAL2-IM-1385-0001.jpeg



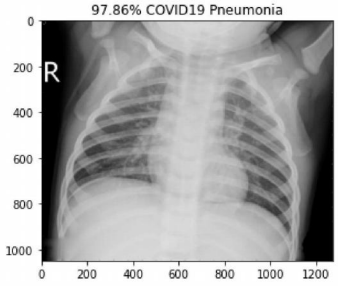
normal\NORMAL2-IM-1396-0001.jpeg



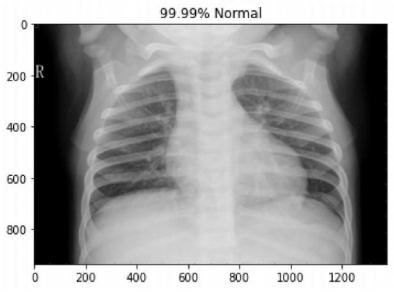
normal\NORMAL2-IM-1400-0001.jpeg



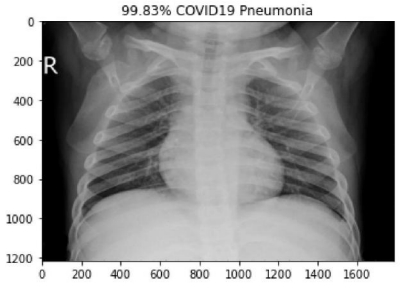
normal\NORMAL2-IM-1401-0001.jpeg



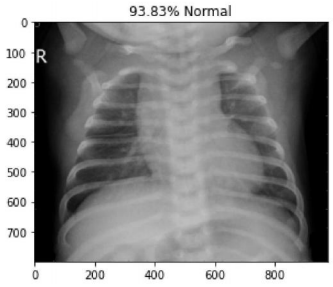
normal\NORMAL2-IM-1406-0001.jpeg



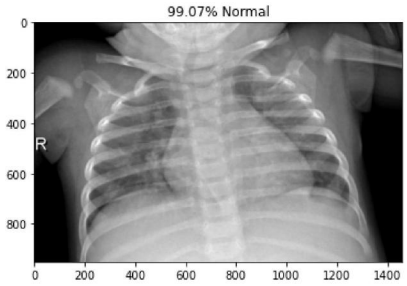
normal\NORMAL2-IM-1412-0001.jpeg



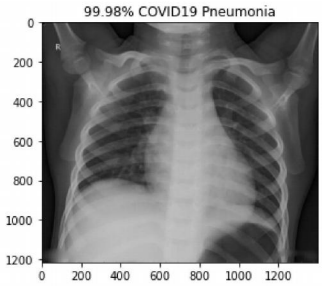
normal\NORMAL2-IM-1419-0001.jpeg



normal\NORMAL2-IM-1422-0001.jpeg



normal\NORMAL2-IM-1423-0001.jpeg



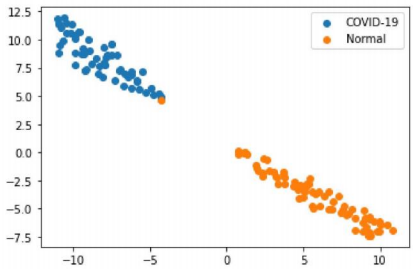
[10 points] TSNE Plot

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 [15]:
```
from sklearn.manifold import TSNE

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

labels = tsne_data_generator.classes
print(tsne_data_generator.class_indices)



X = intermediate_layer_model.predict_generator(tsne_data_generator, verbose=1)
X_embedded = TSNE().fit_transform(X)

classes = ["COVID-19", "Normal"]

for i in range(2):
    clusters = X_embedded[np.where(labels == i)]
    plt.scatter(clusters[:, 0], clusters[:, 1], label = classes[i])

plt.legend()
```

```
Found 130 images belonging to 2 classes.
{'covid': 0, 'normal': 1}
  1/130 [..............................] - ETA: 21s
C:\Users\kpete\AppData\Local\Temp/ipykernel_23760/410331126.py:13: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future version. Please use
`Model.predict`, which supports generators.
  X = intermediate_layer_model.predict_generator(tsne_data_generator, verbose=1)
130/130 [==============================] - 10s 76ms/step
```

Out[15]:
```
<matplotlib.legend.Legend at 0x218bc76ddf0>
```



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

Task 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 [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]:
```
'2.7.0'
```

Load Image Data

In [2]:
```
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

In [3]:
```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center = True,
                                   featurewise_std_normalization = True,width_shift_range=0.2,
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0.1,
                                   zca_whitening = True,channel_shift_range = 20,
                                   horizontal_flip = True,vertical_flip = True,
                                   validation_split = 0.2,fill_mode='constant')


train_batches = train_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                  shuffle=True,batch_size=BATCH_SIZE,
                                                  subset = "training",seed=42,
                                                  class_mode="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.
C:\Users\kpete\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.py:342: UserWarning: This ImageDataGenerator specifies `zca_whitening` which overr
ides 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.

In [4]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, DenseFeatures, Dropout, Conv2D, MaxPool2D, Flatten
```

```
vggmodel = tf.keras.applications.vgg16.VGG16(
    weights="imagenet",
    input_shape=(224, 224, 3),
    include_top=False
)
vggmodel.trainable = False

type(vggmodel)

model = Sequential(
    [
        vggmodel,
        Flatten(),
        Dense(units = 256,activation='relu', name = 'feature_dense'),
        Dense(units = 4, activation = 'softmax', name = 'dense_1')
    ]
)

opt = tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE)
model.compile(optimizer = opt, loss=tf.keras.losses.BinaryCrossentropy(), metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 feature_dense (Dense)       (None, 256)               6422784

 dense_1 (Dense)             (None, 4)                 1028

=================================================================
Total params: 21,138,500
Trainable params: 6,423,812
Non-trainable params: 14,714,688
_________________________________________________________________
```

[5 points] Train Model

In [5]:

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

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

history = model.fit_generator(steps_per_epoch = STEP_SIZE_TRAIN, generator = train_batches, validation_data = valid_batches, validation_steps = STEP_SIZE_VALID, epochs=115
)
```

```
22
6
C:\Users\kpete\AppData\Local\Temp/ipykernel_14888/2354724368.py:7: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Mod
el.fit`, which supports generators.
  history = model.fit_generator(steps_per_epoch = STEP_SIZE_TRAIN, generator = train_batches, validation_data = valid_batches, validation_steps = STEP_SIZE_VALID, epochs=11
5
C:\Users\kpete\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.py:720: UserWarning: This ImageDataGenerator specifies `featurewise_center`, but i
t hasn't been fit on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
C:\Users\kpete\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.py:739: UserWarning: This ImageDataGenerator specifies `zca_whitening`, but it has
n't been fit on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
Epoch 1/115
21/21 [==============================] - 35s 2s/step - loss: 0.5787 - accuracy: 0.3544 - val_loss: 0.5472 - val_accuracy: 0.4000
Epoch 2/115
21/21 [==============================] - 32s 2s/step - loss: 0.4862 - accuracy: 0.4951 - val_loss: 0.4437 - val_accuracy: 0.5600
Epoch 3/115
21/21 [==============================] - 33s 2s/step - loss: 0.4641 - accuracy: 0.4757 - val_loss: 0.4340 - val_accuracy: 0.6000
Epoch 4/115
21/21 [==============================] - 33s 2s/step - loss: 0.4398 - accuracy: 0.5485 - val_loss: 0.4416 - val_accuracy: 0.5000
Epoch 5/115
21/21 [==============================] - 33s 2s/step - loss: 0.4235 - accuracy: 0.6068 - val_loss: 0.4346 - val_accuracy: 0.5600
Epoch 6/115
21/21 [==============================] - 32s 2s/step - loss: 0.4100 - accuracy: 0.5922 - val_loss: 0.4048 - val_accuracy: 0.5400
Epoch 7/115
21/21 [==============================] - 33s 2s/step - loss: 0.3983 - accuracy: 0.6214 - val_loss: 0.4100 - val_accuracy: 0.6200
Epoch 8/115
21/21 [==============================] - 31s 1s/step - loss: 0.4019 - accuracy: 0.5922 - val_loss: 0.3694 - val_accuracy: 0.6400
Epoch 9/115
21/21 [==============================] - 17s 784ms/step - loss: 0.3588 - accuracy: 0.6942 - val_loss: 0.4114 - val_accuracy: 0.5800
Epoch 10/115
21/21 [==============================] - 17s 795ms/step - loss: 0.3682 - accuracy: 0.6810 - val_loss: 0.3648 - val_accuracy: 0.5200
Epoch 11/115
21/21 [==============================] - 17s 786ms/step - loss: 0.3720 - accuracy: 0.6311 - val_loss: 0.4536 - val_accuracy: 0.5200
Epoch 12/115
21/21 [==============================] - 16s 784ms/step - loss: 0.3656 - accuracy: 0.6214 - val_loss: 0.3436 - val_accuracy: 0.6200
Epoch 13/115
21/21 [==============================] - 17s 793ms/step - loss: 0.3405 - accuracy: 0.6990 - val_loss: 0.4215 - val_accuracy: 0.5200
Epoch 14/115
21/21 [==============================] - 17s 789ms/step - loss: 0.3490 - accuracy: 0.6845 - val_loss: 0.3581 - val_accuracy: 0.5600
Epoch 15/115
21/21 [==============================] - 16s 785ms/step - loss: 0.3234 - accuracy: 0.7136 - val_loss: 0.3121 - val_accuracy: 0.6600
Epoch 16/115
21/21 [==============================] - 17s 814ms/step - loss: 0.3117 - accuracy: 0.7282 - val_loss: 0.3373 - val_accuracy: 0.6400
Epoch 17/115
21/21 [==============================] - 17s 807ms/step - loss: 0.3424 - accuracy: 0.6699 - val_loss: 0.3342 - val_accuracy: 0.7000
Epoch 18/115
21/21 [==============================] - 17s 804ms/step - loss: 0.3299 - accuracy: 0.6942 - val_loss: 0.3348 - val_accuracy: 0.6600
Epoch 19/115
21/21 [==============================] - 17s 809ms/step - loss: 0.3321 - accuracy: 0.7238 - val_loss: 0.3485 - val_accuracy: 0.6000
Epoch 20/115
21/21 [==============================] - 17s 796ms/step - loss: 0.3098 - accuracy: 0.6845 - val_loss: 0.3333 - val_accuracy: 0.6800
Epoch 21/115
21/21 [==============================] - 17s 798ms/step - loss: 0.3110 - accuracy: 0.7379 - val_loss: 0.2871 - val_accuracy: 0.8200
Epoch 22/115
```

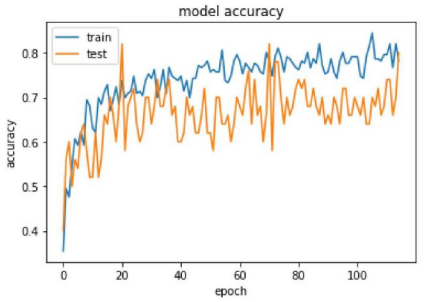
```
21/21 [==============================] - 17s 792ms/step - loss: 0.3233 - accuracy: 0.6990 - val_loss: 0.3514 - val_accuracy: 0.5800
Epoch 23/115
21/21 [==============================] - 17s 812ms/step - loss: 0.3404 - accuracy: 0.7087 - val_loss: 0.3430 - val_accuracy: 0.6800
Epoch 24/115
21/21 [==============================] - 17s 815ms/step - loss: 0.3019 - accuracy: 0.7136 - val_loss: 0.3426 - val_accuracy: 0.7000
Epoch 25/115
21/21 [==============================] - 17s 830ms/step - loss: 0.2968 - accuracy: 0.7476 - val_loss: 0.2731 - val_accuracy: 0.7200
Epoch 26/115
21/21 [==============================] - 18s 837ms/step - loss: 0.3109 - accuracy: 0.7087 - val_loss: 0.3077 - val_accuracy: 0.6400
Epoch 27/115
21/21 [==============================] - 17s 805ms/step - loss: 0.3113 - accuracy: 0.7136 - val_loss: 0.3674 - val_accuracy: 0.6000
Epoch 28/115
21/21 [==============================] - 17s 802ms/step - loss: 0.3085 - accuracy: 0.7039 - val_loss: 0.3366 - val_accuracy: 0.6200
Epoch 29/115
21/21 [==============================] - 17s 802ms/step - loss: 0.3169 - accuracy: 0.7379 - val_loss: 0.3146 - val_accuracy: 0.7000
Epoch 30/115
21/21 [==============================] - 17s 805ms/step - loss: 0.2979 - accuracy: 0.7524 - val_loss: 0.3141 - val_accuracy: 0.7000
Epoch 31/115
21/21 [==============================] - 17s 809ms/step - loss: 0.2980 - accuracy: 0.7427 - val_loss: 0.3212 - val_accuracy: 0.6400
Epoch 32/115
21/21 [==============================] - 17s 822ms/step - loss: 0.2720 - accuracy: 0.7621 - val_loss: 0.3214 - val_accuracy: 0.6800
Epoch 33/115
21/21 [==============================] - 17s 798ms/step - loss: 0.2925 - accuracy: 0.6990 - val_loss: 0.3050 - val_accuracy: 0.7400
Epoch 34/115
21/21 [==============================] - 17s 784ms/step - loss: 0.3154 - accuracy: 0.7233 - val_loss: 0.3129 - val_accuracy: 0.6800
Epoch 35/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2709 - accuracy: 0.7621 - val_loss: 0.3429 - val_accuracy: 0.6800
Epoch 36/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2880 - accuracy: 0.7087 - val_loss: 0.3454 - val_accuracy: 0.7200
Epoch 37/115
21/21 [==============================] - 17s 796ms/step - loss: 0.2685 - accuracy: 0.7670 - val_loss: 0.2865 - val_accuracy: 0.7400
Epoch 38/115
21/21 [==============================] - 17s 803ms/step - loss: 0.2794 - accuracy: 0.7476 - val_loss: 0.3845 - val_accuracy: 0.6600
Epoch 39/115
21/21 [==============================] - 17s 827ms/step - loss: 0.2966 - accuracy: 0.7427 - val_loss: 0.3411 - val_accuracy: 0.6800
Epoch 40/115
21/21 [==============================] - 18s 836ms/step - loss: 0.2876 - accuracy: 0.7379 - val_loss: 0.3348 - val_accuracy: 0.6000
Epoch 41/115
21/21 [==============================] - 17s 810ms/step - loss: 0.2976 - accuracy: 0.7476 - val_loss: 0.3855 - val_accuracy: 0.6000
Epoch 42/115
21/21 [==============================] - 17s 815ms/step - loss: 0.2746 - accuracy: 0.7143 - val_loss: 0.3556 - val_accuracy: 0.6200
Epoch 43/115
21/21 [==============================] - 17s 792ms/step - loss: 0.2752 - accuracy: 0.7379 - val_loss: 0.3032 - val_accuracy: 0.7000
Epoch 44/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2870 - accuracy: 0.6990 - val_loss: 0.3535 - val_accuracy: 0.6600
Epoch 45/115
21/21 [==============================] - 17s 801ms/step - loss: 0.2879 - accuracy: 0.7427 - val_loss: 0.3311 - val_accuracy: 0.6800
Epoch 46/115
21/21 [==============================] - 17s 795ms/step - loss: 0.2959 - accuracy: 0.7427 - val_loss: 0.3693 - val_accuracy: 0.6200
Epoch 47/115
21/21 [==============================] - 17s 836ms/step - loss: 0.2735 - accuracy: 0.7718 - val_loss: 0.3264 - val_accuracy: 0.6200
Epoch 48/115
21/21 [==============================] - 17s 810ms/step - loss: 0.2694 - accuracy: 0.7670 - val_loss: 0.3000 - val_accuracy: 0.6600
Epoch 49/115
21/21 [==============================] - 17s 806ms/step - loss: 0.2544 - accuracy: 0.7718 - val_loss: 0.2897 - val_accuracy: 0.7200
Epoch 50/115
21/21 [==============================] - 17s 810ms/step - loss: 0.2592 - accuracy: 0.7816 - val_loss: 0.3463 - val_accuracy: 0.6200
Epoch 51/115
21/21 [==============================] - 16s 781ms/step - loss: 0.2683 - accuracy: 0.7573 - val_loss: 0.3521 - val_accuracy: 0.6200
Epoch 52/115
21/21 [==============================] - 17s 788ms/step - loss: 0.2581 - accuracy: 0.7621 - val_loss: 0.3392 - val_accuracy: 0.5800
Epoch 53/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2784 - accuracy: 0.7573 - val_loss: 0.3120 - val_accuracy: 0.7000
Epoch 54/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2567 - accuracy: 0.7573 - val_loss: 0.2869 - val_accuracy: 0.7000
Epoch 55/115
21/21 [==============================] - 17s 802ms/step - loss: 0.2559 - accuracy: 0.8058 - val_loss: 0.3142 - val_accuracy: 0.6400
Epoch 56/115
21/21 [==============================] - 17s 831ms/step - loss: 0.2779 - accuracy: 0.7379 - val_loss: 0.3769 - val_accuracy: 0.6400
Epoch 57/115
21/21 [==============================] - 17s 824ms/step - loss: 0.2658 - accuracy: 0.7330 - val_loss: 0.3199 - val_accuracy: 0.6600
Epoch 58/115
21/21 [==============================] - 18s 837ms/step - loss: 0.2664 - accuracy: 0.7476 - val_loss: 0.3554 - val_accuracy: 0.6000
Epoch 59/115
21/21 [==============================] - 17s 791ms/step - loss: 0.2557 - accuracy: 0.7816 - val_loss: 0.3455 - val_accuracy: 0.6400
Epoch 60/115
21/21 [==============================] - 17s 788ms/step - loss: 0.2565 - accuracy: 0.7961 - val_loss: 0.3253 - val_accuracy: 0.7000
Epoch 61/115
21/21 [==============================] - 17s 787ms/step - loss: 0.2587 - accuracy: 0.7816 - val_loss: 0.3100 - val_accuracy: 0.6800
Epoch 62/115
21/21 [==============================] - 17s 798ms/step - loss: 0.2711 - accuracy: 0.7524 - val_loss: 0.3178 - val_accuracy: 0.6600
Epoch 63/115
21/21 [==============================] - 17s 791ms/step - loss: 0.2664 - accuracy: 0.7767 - val_loss: 0.2809 - val_accuracy: 0.7200
Epoch 64/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2713 - accuracy: 0.7670 - val_loss: 0.2823 - val_accuracy: 0.7600
Epoch 65/115
21/21 [==============================] - 17s 801ms/step - loss: 0.2611 - accuracy: 0.7573 - val_loss: 0.3224 - val_accuracy: 0.6400
Epoch 66/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2524 - accuracy: 0.7767 - val_loss: 0.3097 - val_accuracy: 0.7400
Epoch 67/115
21/21 [==============================] - 17s 796ms/step - loss: 0.2581 - accuracy: 0.7718 - val_loss: 0.3135 - val_accuracy: 0.6600
Epoch 68/115
21/21 [==============================] - 17s 792ms/step - loss: 0.2744 - accuracy: 0.7573 - val_loss: 0.3217 - val_accuracy: 0.6800
Epoch 69/115
21/21 [==============================] - 17s 790ms/step - loss: 0.2512 - accuracy: 0.7524 - val_loss: 0.3605 - val_accuracy: 0.6000
Epoch 70/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2601 - accuracy: 0.8010 - val_loss: 0.3053 - val_accuracy: 0.6600
Epoch 71/115
21/21 [==============================] - 17s 800ms/step - loss: 0.2659 - accuracy: 0.7810 - val_loss: 0.2756 - val_accuracy: 0.8200
Epoch 72/115
21/21 [==============================] - 17s 803ms/step - loss: 0.2634 - accuracy: 0.7476 - val_loss: 0.3187 - val_accuracy: 0.5800
Epoch 73/115
21/21 [==============================] - 17s 788ms/step - loss: 0.2396 - accuracy: 0.7913 - val_loss: 0.2673 - val_accuracy: 0.7800
Epoch 74/115
21/21 [==============================] - 17s 791ms/step - loss: 0.2519 - accuracy: 0.8107 - val_loss: 0.2766 - val_accuracy: 0.7800
Epoch 75/115
21/21 [==============================] - 17s 799ms/step - loss: 0.2586 - accuracy: 0.7905 - val_loss: 0.3119 - val_accuracy: 0.7000
Epoch 76/115
21/21 [==============================] - 17s 791ms/step - loss: 0.2653 - accuracy: 0.7573 - val_loss: 0.3069 - val_accuracy: 0.6400
```

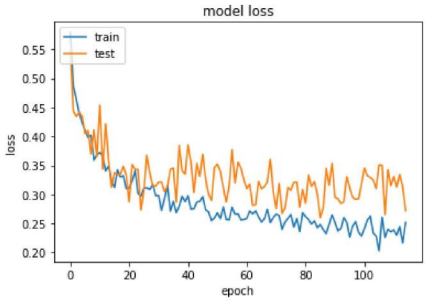
```
Epoch 77/115
21/21 [==============================] - 17s 795ms/step - loss: 0.2435 - accuracy: 0.7913 - val_loss: 0.3210 - val_accuracy: 0.7000
Epoch 78/115
21/21 [==============================] - 17s 787ms/step - loss: 0.2581 - accuracy: 0.7864 - val_loss: 0.3212 - val_accuracy: 0.6600
Epoch 79/115
21/21 [==============================] - 17s 785ms/step - loss: 0.2360 - accuracy: 0.7767 - val_loss: 0.2778 - val_accuracy: 0.6800
Epoch 80/115
21/21 [==============================] - 16s 798ms/step - loss: 0.2683 - accuracy: 0.7670 - val_loss: 0.3082 - val_accuracy: 0.7200
Epoch 81/115
21/21 [==============================] - 17s 786ms/step - loss: 0.2615 - accuracy: 0.7621 - val_loss: 0.2845 - val_accuracy: 0.7400
Epoch 82/115
21/21 [==============================] - 17s 790ms/step - loss: 0.2564 - accuracy: 0.7816 - val_loss: 0.3335 - val_accuracy: 0.7200
Epoch 83/115
21/21 [==============================] - 17s 791ms/step - loss: 0.2489 - accuracy: 0.7767 - val_loss: 0.3140 - val_accuracy: 0.7400
Epoch 84/115
21/21 [==============================] - 17s 792ms/step - loss: 0.2541 - accuracy: 0.8010 - val_loss: 0.3220 - val_accuracy: 0.6800
Epoch 85/115
21/21 [==============================] - 17s 814ms/step - loss: 0.2423 - accuracy: 0.7667 - val_loss: 0.2993 - val_accuracy: 0.6800
Epoch 86/115
21/21 [==============================] - 16s 785ms/step - loss: 0.2481 - accuracy: 0.7864 - val_loss: 0.2596 - val_accuracy: 0.7200
Epoch 87/115
21/21 [==============================] - 17s 786ms/step - loss: 0.2390 - accuracy: 0.7767 - val_loss: 0.2769 - val_accuracy: 0.6800
Epoch 88/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2322 - accuracy: 0.8204 - val_loss: 0.3449 - val_accuracy: 0.6600
Epoch 89/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2485 - accuracy: 0.7718 - val_loss: 0.3154 - val_accuracy: 0.7000
Epoch 90/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2644 - accuracy: 0.7524 - val_loss: 0.3531 - val_accuracy: 0.6400
Epoch 91/115
21/21 [==============================] - 17s 810ms/step - loss: 0.2518 - accuracy: 0.7573 - val_loss: 0.2949 - val_accuracy: 0.6600
Epoch 92/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2370 - accuracy: 0.7864 - val_loss: 0.2924 - val_accuracy: 0.6400
Epoch 93/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2412 - accuracy: 0.7573 - val_loss: 0.2843 - val_accuracy: 0.7000
Epoch 94/115
21/21 [==============================] - 17s 788ms/step - loss: 0.2600 - accuracy: 0.7427 - val_loss: 0.2872 - val_accuracy: 0.6800
Epoch 95/115
21/21 [==============================] - 17s 816ms/step - loss: 0.2518 - accuracy: 0.7816 - val_loss: 0.3305 - val_accuracy: 0.6400
Epoch 96/115
21/21 [==============================] - 17s 786ms/step - loss: 0.2264 - accuracy: 0.8010 - val_loss: 0.3109 - val_accuracy: 0.7200
Epoch 97/115
21/21 [==============================] - 16s 777ms/step - loss: 0.2450 - accuracy: 0.7767 - val_loss: 0.2951 - val_accuracy: 0.7200
Epoch 98/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2528 - accuracy: 0.7767 - val_loss: 0.2913 - val_accuracy: 0.6600
Epoch 99/115
21/21 [==============================] - 17s 824ms/step - loss: 0.2348 - accuracy: 0.7913 - val_loss: 0.2924 - val_accuracy: 0.6600
Epoch 100/115
21/21 [==============================] - 17s 817ms/step - loss: 0.2278 - accuracy: 0.7913 - val_loss: 0.3180 - val_accuracy: 0.7000
Epoch 101/115
21/21 [==============================] - 17s 796ms/step - loss: 0.2411 - accuracy: 0.7913 - val_loss: 0.3452 - val_accuracy: 0.6800
Epoch 102/115
21/21 [==============================] - 17s 790ms/step - loss: 0.2556 - accuracy: 0.7476 - val_loss: 0.3322 - val_accuracy: 0.6600
Epoch 103/115
21/21 [==============================] - 17s 820ms/step - loss: 0.2624 - accuracy: 0.7427 - val_loss: 0.3292 - val_accuracy: 0.7000
Epoch 104/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2334 - accuracy: 0.7913 - val_loss: 0.3250 - val_accuracy: 0.6400
Epoch 105/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2278 - accuracy: 0.8155 - val_loss: 0.3099 - val_accuracy: 0.6400
Epoch 106/115
21/21 [==============================] - 17s 790ms/step - loss: 0.2028 - accuracy: 0.8447 - val_loss: 0.3507 - val_accuracy: 0.7000
Epoch 107/115
21/21 [==============================] - 17s 787ms/step - loss: 0.2607 - accuracy: 0.7864 - val_loss: 0.3501 - val_accuracy: 0.6800
Epoch 108/115
21/21 [==============================] - 17s 791ms/step - loss: 0.2258 - accuracy: 0.7864 - val_loss: 0.2652 - val_accuracy: 0.7200
Epoch 109/115
21/21 [==============================] - 17s 787ms/step - loss: 0.2396 - accuracy: 0.7816 - val_loss: 0.3425 - val_accuracy: 0.6600
Epoch 110/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2355 - accuracy: 0.7961 - val_loss: 0.3151 - val_accuracy: 0.6800
Epoch 111/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2388 - accuracy: 0.7961 - val_loss: 0.3310 - val_accuracy: 0.7400
Epoch 112/115
21/21 [==============================] - 17s 799ms/step - loss: 0.2296 - accuracy: 0.8204 - val_loss: 0.3125 - val_accuracy: 0.7400
Epoch 113/115
21/21 [==============================] - 17s 791ms/step - loss: 0.2445 - accuracy: 0.7670 - val_loss: 0.3343 - val_accuracy: 0.6600
Epoch 114/115
21/21 [==============================] - 17s 794ms/step - loss: 0.2163 - accuracy: 0.8204 - val_loss: 0.3131 - val_accuracy: 0.7000
Epoch 115/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2515 - accuracy: 0.7816 - val_loss: 0.2719 - val_accuracy: 0.8000
```

[5 points] Plot Accuracy and Loss During Training

In [6]:

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()


plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Testing Model

In [7]:
```
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:' , x[0])
print('Test accuracy:',x[1])
```

```
Found 36 images belonging to 4 classes.
36
 2/36 [>.............................] - ETA: 2s - loss: 1.3187 - accuracy: 0.5000
C:\Users\kpete\AppData\Local\Temp/ipykernel_14888/463003407.py:7: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use
`Model.evaluate`, which supports generators.
  x = model.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
36/36 [==============================] - 3s 80ms/step - loss: 0.3132 - accuracy: 0.7222
Test loss: 0.3131543695926666
Test accuracy: 0.7222222089767456
```

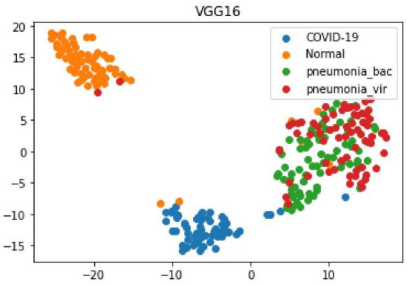
[10 points] TSNE Plot

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 [8]:
```
from sklearn.manifold import TSNE

iLM = tf.keras.models.Model(inputs=model.input, outputs=model.get_layer('feature_dense').output)

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

tEG.reset()

classifications = tEG.classes

print(tEG.class_indices)

a = iLM.predict_generator(tEG, verbose=1)

a2 = TSNE().fit_transform(a)

legend = ["COVID-19", "Normal", "pneumonia_bac", "pneumonia_vir"]

plt.title('VGG16')
for i in range(4):
    clusters = a2[np.where(i==classifications)]
    plt.scatter(clusters[:, 0], clusters[:, 1], label = legend[i])

plt.legend()
```

```
Found 270 images belonging to 4 classes.
{'covid': 0, 'normal': 1, 'pneumonia_bac': 2, 'pneumonia_vir': 3}
C:\Users\kpete\AppData\Local\Temp/ipykernel_14888/1148479893.py:13: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future version. Please use
`Model.predict`, which supports generators.
  a = iLM.predict_generator(tEG, verbose=1)
270/270 [==============================] - 20s 74ms/step
```

Out[8]:
```
<matplotlib.legend.Legend at 0x1de521aa520>
```



In [9]:

```
alexnet = Sequential(
    [

        Conv2D(filters=96, input_shape=(224,224,3),kernel_size=(11,11), strides=(4,4), padding='valid', activation='relu'),
        MaxPool2D(pool_size=(2,2), strides=(2,2), padding='valid'),

        Conv2D(filters=256, kernel_size=(11,11), strides=(1,1), padding='valid', activation='relu'),
        MaxPool2D(pool_size=(2,2), strides=(2,2), padding='valid'),

        Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'),

        Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'),

        Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='valid', activation='relu'),
        MaxPool2D(pool_size=(2,2), strides=(2,2), padding='valid'),

        Flatten(),
        Dense(256, activation='relu', name='feature_dense'),

        Dropout(.4),

        Dense(4096, activation = 'relu'),
        Dropout(.4),

        Dense(1000, activation = 'relu'),
        Dropout(.4),

        Dense(4, activation='softmax')
    ]
)

alexnet.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 54, 54, 96)        34944

 max_pooling2d (MaxPooling2D  (None, 27, 27, 96)       0
 )

 conv2d_1 (Conv2D)           (None, 17, 17, 256)       2973952

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 256)        0
 2D)

 conv2d_2 (Conv2D)           (None, 6, 6, 384)         885120

 conv2d_3 (Conv2D)           (None, 4, 4, 384)         1327488

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         884992

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 flatten_1 (Flatten)         (None, 256)               0

 feature_dense (Dense)       (None, 256)               65792

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 4096)              1052672

 dropout_1 (Dropout)         (None, 4096)              0

 dense_1 (Dense)             (None, 1000)              4097000

 dropout_2 (Dropout)         (None, 1000)              0

 dense_2 (Dense)             (None, 4)                 4004

=================================================================
Total params: 11,325,964
Trainable params: 11,325,964
Non-trainable params: 0
_________________________________________________________________
```

In [11]:

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

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

history = model.fit_generator(steps_per_epoch = STEP_SIZE_TRAIN, generator = train_batches, validation_data = valid_batches, validation_steps = STEP_SIZE_VALID, epochs=115)
```

```
22
6
Epoch 1/115
```

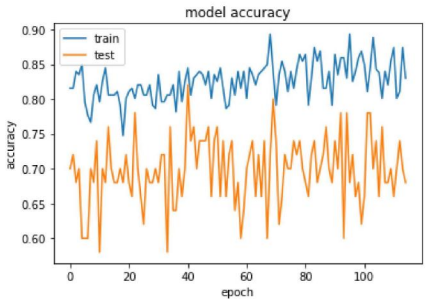
```
C:\Users\kpete\AppData\Local\Temp/ipykernel_14888/693659470.py:7: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Mode
l.fit`, which supports generators.
  history = model.fit_generator(steps_per_epoch = STEP_SIZE_TRAIN, generator = train_batches, validation_data = valid_batches, validation_steps = STEP_SIZE_VALID, epochs=11
5)
21/21 [==============================] - 17s 794ms/step - loss: 0.2179 - accuracy: 0.8155 - val_loss: 0.2627 - val_accuracy: 0.7000
Epoch 2/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2206 - accuracy: 0.8155 - val_loss: 0.2658 - val_accuracy: 0.7200
Epoch 3/115
21/21 [==============================] - 17s 794ms/step - loss: 0.2045 - accuracy: 0.8398 - val_loss: 0.2884 - val_accuracy: 0.6800
Epoch 4/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2082 - accuracy: 0.8350 - val_loss: 0.3265 - val_accuracy: 0.7000
Epoch 5/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2150 - accuracy: 0.8495 - val_loss: 0.3604 - val_accuracy: 0.6000
Epoch 6/115
21/21 [==============================] - 17s 798ms/step - loss: 0.2379 - accuracy: 0.7961 - val_loss: 0.3384 - val_accuracy: 0.6000
Epoch 7/115
21/21 [==============================] - 17s 797ms/step - loss: 0.2345 - accuracy: 0.7767 - val_loss: 0.3383 - val_accuracy: 0.6000
Epoch 8/115
21/21 [==============================] - 17s 806ms/step - loss: 0.2428 - accuracy: 0.7667 - val_loss: 0.2896 - val_accuracy: 0.7000
Epoch 9/115
21/21 [==============================] - 16s 784ms/step - loss: 0.2278 - accuracy: 0.8058 - val_loss: 0.2962 - val_accuracy: 0.6800
Epoch 10/115
21/21 [==============================] - 17s 787ms/step - loss: 0.2234 - accuracy: 0.8204 - val_loss: 0.2749 - val_accuracy: 0.7400
Epoch 11/115
21/21 [==============================] - 16s 782ms/step - loss: 0.2305 - accuracy: 0.7961 - val_loss: 0.3882 - val_accuracy: 0.5800
Epoch 12/115
21/21 [==============================] - 17s 796ms/step - loss: 0.2037 - accuracy: 0.8252 - val_loss: 0.2732 - val_accuracy: 0.7000
Epoch 13/115
21/21 [==============================] - 17s 802ms/step - loss: 0.1979 - accuracy: 0.8447 - val_loss: 0.3195 - val_accuracy: 0.6800
Epoch 14/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2161 - accuracy: 0.8058 - val_loss: 0.2649 - val_accuracy: 0.7600
Epoch 15/115
21/21 [==============================] - 17s 804ms/step - loss: 0.2243 - accuracy: 0.8058 - val_loss: 0.2823 - val_accuracy: 0.7000
Epoch 16/115
21/21 [==============================] - 17s 792ms/step - loss: 0.2148 - accuracy: 0.8058 - val_loss: 0.3037 - val_accuracy: 0.6800
Epoch 17/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2149 - accuracy: 0.8107 - val_loss: 0.3154 - val_accuracy: 0.6800
Epoch 18/115
21/21 [==============================] - 17s 802ms/step - loss: 0.2290 - accuracy: 0.7913 - val_loss: 0.3005 - val_accuracy: 0.7000
Epoch 19/115
21/21 [==============================] - 17s 801ms/step - loss: 0.2456 - accuracy: 0.7476 - val_loss: 0.2929 - val_accuracy: 0.6800
Epoch 20/115
21/21 [==============================] - 17s 822ms/step - loss: 0.1959 - accuracy: 0.8010 - val_loss: 0.3274 - val_accuracy: 0.7200
Epoch 21/115
21/21 [==============================] - 17s 833ms/step - loss: 0.2063 - accuracy: 0.8107 - val_loss: 0.3296 - val_accuracy: 0.6800
Epoch 22/115
21/21 [==============================] - 17s 808ms/step - loss: 0.2416 - accuracy: 0.8155 - val_loss: 0.3301 - val_accuracy: 0.6600
Epoch 23/115
21/21 [==============================] - 17s 823ms/step - loss: 0.2100 - accuracy: 0.8010 - val_loss: 0.2559 - val_accuracy: 0.7800
Epoch 24/115
21/21 [==============================] - 17s 796ms/step - loss: 0.2130 - accuracy: 0.8204 - val_loss: 0.2878 - val_accuracy: 0.7000
Epoch 25/115
21/21 [==============================] - 17s 794ms/step - loss: 0.2131 - accuracy: 0.8204 - val_loss: 0.3743 - val_accuracy: 0.6600
Epoch 26/115
21/21 [==============================] - 17s 798ms/step - loss: 0.2341 - accuracy: 0.8058 - val_loss: 0.3105 - val_accuracy: 0.6200
Epoch 27/115
21/21 [==============================] - 17s 794ms/step - loss: 0.2159 - accuracy: 0.8058 - val_loss: 0.2986 - val_accuracy: 0.7000
Epoch 28/115
21/21 [==============================] - 17s 790ms/step - loss: 0.1974 - accuracy: 0.8204 - val_loss: 0.3351 - val_accuracy: 0.6800
Epoch 29/115
21/21 [==============================] - 17s 785ms/step - loss: 0.2238 - accuracy: 0.7913 - val_loss: 0.2895 - val_accuracy: 0.6800
Epoch 30/115
21/21 [==============================] - 17s 802ms/step - loss: 0.2188 - accuracy: 0.7864 - val_loss: 0.3218 - val_accuracy: 0.7000
Epoch 31/115
21/21 [==============================] - 17s 804ms/step - loss: 0.2047 - accuracy: 0.8350 - val_loss: 0.2772 - val_accuracy: 0.6800
Epoch 32/115
21/21 [==============================] - 17s 799ms/step - loss: 0.2237 - accuracy: 0.7961 - val_loss: 0.2809 - val_accuracy: 0.7200
Epoch 33/115
21/21 [==============================] - 17s 812ms/step - loss: 0.2137 - accuracy: 0.7961 - val_loss: 0.2765 - val_accuracy: 0.7200
Epoch 34/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2129 - accuracy: 0.8058 - val_loss: 0.4364 - val_accuracy: 0.5800
Epoch 35/115
21/21 [==============================] - 17s 796ms/step - loss: 0.2109 - accuracy: 0.8058 - val_loss: 0.3088 - val_accuracy: 0.7600
Epoch 36/115
21/21 [==============================] - 17s 805ms/step - loss: 0.2067 - accuracy: 0.8204 - val_loss: 0.3292 - val_accuracy: 0.6400
Epoch 37/115
21/21 [==============================] - 17s 792ms/step - loss: 0.2280 - accuracy: 0.7816 - val_loss: 0.3863 - val_accuracy: 0.6400
Epoch 38/115
21/21 [==============================] - 17s 808ms/step - loss: 0.1974 - accuracy: 0.8398 - val_loss: 0.3556 - val_accuracy: 0.7000
Epoch 39/115
21/21 [==============================] - 17s 789ms/step - loss: 0.2061 - accuracy: 0.7961 - val_loss: 0.3782 - val_accuracy: 0.6600
Epoch 40/115
21/21 [==============================] - 16s 798ms/step - loss: 0.2101 - accuracy: 0.8252 - val_loss: 0.3006 - val_accuracy: 0.7000
Epoch 41/115
21/21 [==============================] - 17s 796ms/step - loss: 0.2033 - accuracy: 0.8447 - val_loss: 0.2702 - val_accuracy: 0.8200
Epoch 42/115
21/21 [==============================] - 17s 803ms/step - loss: 0.2120 - accuracy: 0.8058 - val_loss: 0.2749 - val_accuracy: 0.7400
Epoch 43/115
21/21 [==============================] - 17s 792ms/step - loss: 0.1936 - accuracy: 0.8301 - val_loss: 0.2798 - val_accuracy: 0.7600
Epoch 44/115
21/21 [==============================] - 17s 797ms/step - loss: 0.1972 - accuracy: 0.8350 - val_loss: 0.2703 - val_accuracy: 0.7000
Epoch 45/115
21/21 [==============================] - 17s 791ms/step - loss: 0.1908 - accuracy: 0.8398 - val_loss: 0.2435 - val_accuracy: 0.7400
Epoch 46/115
21/21 [==============================] - 17s 787ms/step - loss: 0.2036 - accuracy: 0.8350 - val_loss: 0.2515 - val_accuracy: 0.7400
Epoch 47/115
21/21 [==============================] - 17s 795ms/step - loss: 0.2130 - accuracy: 0.8204 - val_loss: 0.3026 - val_accuracy: 0.7400
Epoch 48/115
21/21 [==============================] - 17s 816ms/step - loss: 0.2009 - accuracy: 0.8398 - val_loss: 0.2656 - val_accuracy: 0.7600
Epoch 49/115
21/21 [==============================] - 17s 793ms/step - loss: 0.2033 - accuracy: 0.8010 - val_loss: 0.3232 - val_accuracy: 0.6600
Epoch 50/115
21/21 [==============================] - 17s 792ms/step - loss: 0.1897 - accuracy: 0.8350 - val_loss: 0.2956 - val_accuracy: 0.7400
Epoch 51/115
21/21 [==============================] - 18s 862ms/step - loss: 0.2122 - accuracy: 0.8252 - val_loss: 0.2601 - val_accuracy: 0.7600
Epoch 52/115
21/21 [==============================] - 18s 863ms/step - loss: 0.1758 - accuracy: 0.8447 - val_loss: 0.3033 - val_accuracy: 0.6600
Epoch 53/115
21/21 [==============================] - 18s 839ms/step - loss: 0.2002 - accuracy: 0.8155 - val_loss: 0.3192 - val_accuracy: 0.7400
```

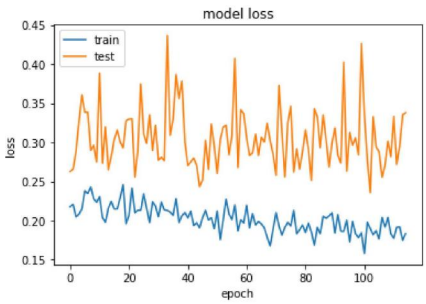
```
Epoch 54/115
21/21 [==============================] - 17s 824ms/step - loss: 0.2271 - accuracy: 0.7864 - val_loss: 0.3218 - val_accuracy: 0.6600
Epoch 55/115
21/21 [==============================] - 19s 890ms/step - loss: 0.2082 - accuracy: 0.7913 - val_loss: 0.2841 - val_accuracy: 0.7200
Epoch 56/115
21/21 [==============================] - 18s 833ms/step - loss: 0.2014 - accuracy: 0.8301 - val_loss: 0.3078 - val_accuracy: 0.7400
Epoch 57/115
21/21 [==============================] - 18s 842ms/step - loss: 0.2198 - accuracy: 0.8058 - val_loss: 0.4072 - val_accuracy: 0.6400
Epoch 58/115
21/21 [==============================] - 18s 843ms/step - loss: 0.1867 - accuracy: 0.8398 - val_loss: 0.2677 - val_accuracy: 0.6800
Epoch 59/115
21/21 [==============================] - 18s 863ms/step - loss: 0.2008 - accuracy: 0.8155 - val_loss: 0.3414 - val_accuracy: 0.6000
Epoch 60/115
21/21 [==============================] - 18s 835ms/step - loss: 0.1970 - accuracy: 0.8398 - val_loss: 0.3365 - val_accuracy: 0.6400
Epoch 61/115
21/21 [==============================] - 18s 838ms/step - loss: 0.2195 - accuracy: 0.8010 - val_loss: 0.3054 - val_accuracy: 0.7000
Epoch 62/115
21/21 [==============================] - 17s 829ms/step - loss: 0.1907 - accuracy: 0.8447 - val_loss: 0.2831 - val_accuracy: 0.7200
Epoch 63/115
21/21 [==============================] - 18s 853ms/step - loss: 0.2088 - accuracy: 0.8350 - val_loss: 0.2864 - val_accuracy: 0.7400
Epoch 64/115
21/21 [==============================] - 18s 834ms/step - loss: 0.1944 - accuracy: 0.8204 - val_loss: 0.3103 - val_accuracy: 0.6600
Epoch 65/115
21/21 [==============================] - 18s 837ms/step - loss: 0.1989 - accuracy: 0.8350 - val_loss: 0.2831 - val_accuracy: 0.7200
Epoch 66/115
21/21 [==============================] - 18s 845ms/step - loss: 0.1957 - accuracy: 0.8398 - val_loss: 0.3062 - val_accuracy: 0.6600
Epoch 67/115
21/21 [==============================] - 18s 844ms/step - loss: 0.1904 - accuracy: 0.8447 - val_loss: 0.3005 - val_accuracy: 0.7400
Epoch 68/115
21/21 [==============================] - 18s 835ms/step - loss: 0.1784 - accuracy: 0.8495 - val_loss: 0.3242 - val_accuracy: 0.6000
Epoch 69/115
21/21 [==============================] - 18s 890ms/step - loss: 0.1676 - accuracy: 0.8932 - val_loss: 0.3015 - val_accuracy: 0.7200
Epoch 70/115
21/21 [==============================] - 19s 889ms/step - loss: 0.1894 - accuracy: 0.8398 - val_loss: 0.2844 - val_accuracy: 0.8000
Epoch 71/115
21/21 [==============================] - 18s 845ms/step - loss: 0.2100 - accuracy: 0.7913 - val_loss: 0.2580 - val_accuracy: 0.7400
Epoch 72/115
21/21 [==============================] - 19s 883ms/step - loss: 0.1925 - accuracy: 0.8350 - val_loss: 0.3726 - val_accuracy: 0.6200
Epoch 73/115
21/21 [==============================] - 17s 821ms/step - loss: 0.1814 - accuracy: 0.8544 - val_loss: 0.3116 - val_accuracy: 0.6600
Epoch 74/115
21/21 [==============================] - 17s 788ms/step - loss: 0.1914 - accuracy: 0.8398 - val_loss: 0.2560 - val_accuracy: 0.7200
Epoch 75/115
21/21 [==============================] - 16s 770ms/step - loss: 0.1977 - accuracy: 0.8107 - val_loss: 0.3256 - val_accuracy: 0.7000
Epoch 76/115
21/21 [==============================] - 16s 773ms/step - loss: 0.1929 - accuracy: 0.8398 - val_loss: 0.3462 - val_accuracy: 0.7000
Epoch 77/115
21/21 [==============================] - 16s 788ms/step - loss: 0.2129 - accuracy: 0.8155 - val_loss: 0.2619 - val_accuracy: 0.7400
Epoch 78/115
21/21 [==============================] - 16s 782ms/step - loss: 0.1831 - accuracy: 0.8429 - val_loss: 0.2914 - val_accuracy: 0.7200
Epoch 79/115
21/21 [==============================] - 16s 781ms/step - loss: 0.1884 - accuracy: 0.8641 - val_loss: 0.2658 - val_accuracy: 0.7400
Epoch 80/115
21/21 [==============================] - 17s 814ms/step - loss: 0.1941 - accuracy: 0.8544 - val_loss: 0.2880 - val_accuracy: 0.7000
Epoch 81/115
21/21 [==============================] - 17s 808ms/step - loss: 0.1848 - accuracy: 0.8641 - val_loss: 0.3156 - val_accuracy: 0.6800
Epoch 82/115
21/21 [==============================] - 17s 807ms/step - loss: 0.1969 - accuracy: 0.7913 - val_loss: 0.2906 - val_accuracy: 0.6600
Epoch 83/115
21/21 [==============================] - 17s 781ms/step - loss: 0.1854 - accuracy: 0.8301 - val_loss: 0.2512 - val_accuracy: 0.7200
Epoch 84/115
21/21 [==============================] - 17s 797ms/step - loss: 0.1688 - accuracy: 0.8738 - val_loss: 0.3427 - val_accuracy: 0.7400
Epoch 85/115
21/21 [==============================] - 16s 788ms/step - loss: 0.1913 - accuracy: 0.8544 - val_loss: 0.3323 - val_accuracy: 0.6800
Epoch 86/115
21/21 [==============================] - 16s 773ms/step - loss: 0.1832 - accuracy: 0.8689 - val_loss: 0.2930 - val_accuracy: 0.7000
Epoch 87/115
21/21 [==============================] - 16s 779ms/step - loss: 0.2055 - accuracy: 0.8155 - val_loss: 0.3350 - val_accuracy: 0.7200
Epoch 88/115
21/21 [==============================] - 16s 793ms/step - loss: 0.2028 - accuracy: 0.8155 - val_loss: 0.3026 - val_accuracy: 0.7600
Epoch 89/115
21/21 [==============================] - 16s 774ms/step - loss: 0.2059 - accuracy: 0.8398 - val_loss: 0.2685 - val_accuracy: 0.7000
Epoch 90/115
21/21 [==============================] - 17s 790ms/step - loss: 0.2097 - accuracy: 0.7913 - val_loss: 0.2994 - val_accuracy: 0.6800
Epoch 91/115
21/21 [==============================] - 16s 779ms/step - loss: 0.1841 - accuracy: 0.8641 - val_loss: 0.3182 - val_accuracy: 0.7400
Epoch 92/115
21/21 [==============================] - 16s 774ms/step - loss: 0.2074 - accuracy: 0.8350 - val_loss: 0.2835 - val_accuracy: 0.7000
Epoch 93/115
21/21 [==============================] - 16s 779ms/step - loss: 0.1865 - accuracy: 0.8592 - val_loss: 0.2735 - val_accuracy: 0.7800
Epoch 94/115
21/21 [==============================] - 16s 777ms/step - loss: 0.1863 - accuracy: 0.8592 - val_loss: 0.4025 - val_accuracy: 0.6000
Epoch 95/115
21/21 [==============================] - 17s 786ms/step - loss: 0.2004 - accuracy: 0.8301 - val_loss: 0.2630 - val_accuracy: 0.7800
Epoch 96/115
21/21 [==============================] - 16s 771ms/step - loss: 0.1730 - accuracy: 0.8932 - val_loss: 0.3125 - val_accuracy: 0.6800
Epoch 97/115
21/21 [==============================] - 16s 783ms/step - loss: 0.1989 - accuracy: 0.8252 - val_loss: 0.2964 - val_accuracy: 0.7200
Epoch 98/115
21/21 [==============================] - 16s 775ms/step - loss: 0.1831 - accuracy: 0.8398 - val_loss: 0.3055 - val_accuracy: 0.6600
Epoch 99/115
21/21 [==============================] - 16s 777ms/step - loss: 0.1781 - accuracy: 0.8592 - val_loss: 0.2840 - val_accuracy: 0.6800
Epoch 100/115
21/21 [==============================] - 16s 781ms/step - loss: 0.1845 - accuracy: 0.8689 - val_loss: 0.4263 - val_accuracy: 0.6200
Epoch 101/115
21/21 [==============================] - 17s 785ms/step - loss: 0.1579 - accuracy: 0.8495 - val_loss: 0.3365 - val_accuracy: 0.6600
Epoch 102/115
21/21 [==============================] - 16s 771ms/step - loss: 0.1977 - accuracy: 0.8107 - val_loss: 0.2772 - val_accuracy: 0.7800
Epoch 103/115
21/21 [==============================] - 16s 774ms/step - loss: 0.1899 - accuracy: 0.8447 - val_loss: 0.2357 - val_accuracy: 0.7800
Epoch 104/115
21/21 [==============================] - 16s 780ms/step - loss: 0.1818 - accuracy: 0.8883 - val_loss: 0.3326 - val_accuracy: 0.7000
Epoch 105/115
21/21 [==============================] - 17s 785ms/step - loss: 0.1871 - accuracy: 0.8429 - val_loss: 0.2939 - val_accuracy: 0.7400
Epoch 106/115
21/21 [==============================] - 17s 786ms/step - loss: 0.1765 - accuracy: 0.8381 - val_loss: 0.2883 - val_accuracy: 0.6600
Epoch 107/115
21/21 [==============================] - 16s 776ms/step - loss: 0.2038 - accuracy: 0.8010 - val_loss: 0.2554 - val_accuracy: 0.7600
Epoch 108/115
```

```
21/21 [==============================] - 17s 788ms/step - loss: 0.1918 - accuracy: 0.8398 - val_loss: 0.2704 - val_accuracy: 0.6800
Epoch 109/115
21/21 [==============================] - 16s 780ms/step - loss: 0.2038 - accuracy: 0.8204 - val_loss: 0.3013 - val_accuracy: 0.6800
Epoch 110/115
21/21 [==============================] - 16s 792ms/step - loss: 0.1829 - accuracy: 0.8544 - val_loss: 0.2817 - val_accuracy: 0.7000
Epoch 111/115
21/21 [==============================] - 16s 781ms/step - loss: 0.1774 - accuracy: 0.8738 - val_loss: 0.3330 - val_accuracy: 0.6600
Epoch 112/115
21/21 [==============================] - 16s 771ms/step - loss: 0.1913 - accuracy: 0.8010 - val_loss: 0.2717 - val_accuracy: 0.7000
Epoch 113/115
21/21 [==============================] - 16s 770ms/step - loss: 0.1919 - accuracy: 0.8107 - val_loss: 0.2948 - val_accuracy: 0.7400
Epoch 114/115
21/21 [==============================] - 16s 772ms/step - loss: 0.1749 - accuracy: 0.8738 - val_loss: 0.3351 - val_accuracy: 0.7000
Epoch 115/115
21/21 [==============================] - 16s 773ms/step - loss: 0.1831 - accuracy: 0.8301 - val_loss: 0.3377 - val_accuracy: 0.6800
```

In [12]:

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()


plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





In [15]:

```
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:' , x[0])
print('Test accuracy:',x[1])
```

```
Found 36 images belonging to 4 classes.
36
 2/36 [>.............................] - ETA: 2s - loss: 2.0855 - accuracy: 0.5000
C:\Users\kpete\AppData\Local\Temp/ipykernel_14888/463003407.py:7: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use
`Model.evaluate`, which supports generators.
  x = model.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
36/36 [==============================] - 3s 80ms/step - loss: 0.4222 - accuracy: 0.6667
Test loss: 0.4221991002559662
Test accuracy: 0.6666666865348816
```

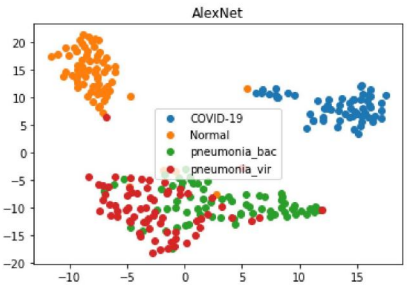
In [16]:

```
from sklearn.manifold import TSNE

iLM = tf.keras.models.Model(inputs=model.input, outputs=model.get_layer('feature_dense').output)

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

tEG.reset()

classifications = tEG.classes

print(tEG.class_indices)

a = iLM.predict_generator(tEG, verbose=1)

a2 = TSNE().fit_transform(a)
```

```
legend = ["COVID-19", "Normal", "pneumonia_bac", "pneumonia_vir"]

plt.title('AlexNet')
for i in range(4):
    clusters = a2[np.where(i==classifications)]
    plt.scatter(clusters[:, 0], clusters[:, 1], label = legend[i])

plt.legend()
```

```
Found 270 images belonging to 4 classes.
{'covid': 0, 'normal': 1, 'pneumonia_bac': 2, 'pneumonia_vir': 3}
  1/270 [..............................] - ETA: 46s
C:\Users\kpete\AppData\Local\Temp/ipykernel_14888/1634499548.py:13: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future version. Please use
`Model.predict`, which supports generators.
  a = iLM.predict_generator(tEG, verbose=1)
270/270 [==============================] - 20s 73ms/step
```

Out[16]: <matplotlib.legend.Legend at 0x1de59a234c0>



In [ ]:

Danny Xu, Kyle Peters
CS440
Class Challenge: Report

- **[30 points]** ***Report***
    - **[5 points]** Describe the architectures used in detail: layers, layer dimensions, dropout layers, etc. for both tasks. List the optimizer, loss function, parameters, and any regularization used in both tasks
    - **[10 points]** Comparison of the performance of different architectures for the second task and relating this to the architecture and parameter settings used
    - **[10 points]** Plot and comment on the accuracy and the loss for both tasks
    - **[5 points]** Plot and comment on the t-SNE visualizations
    - **[Bonus: 5 points]** Run the training on a GPU on the SCC cluster and include a CPU *vs.* GPU training time comparison by taking snapshots from your terminal

TASK 1:

Architecture:
For the Architecture we used the model VGG16.

In this Architecture we used 4 different layers:

1. VGG16 Model Layer
2. Flatten Layer
3. Dense feature Layer
4. Another Dense Layer

Optimizer Task 1: Adam with learning rate 0.001

Loss Function Task 1: Binary_Crossentropy

Parameters Task 1: There are a total of 21,137,729 parameters, which includes 6,423,041 trainable and 14,714,688 untrainable ones.

Regularization Task 1: None

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               6422784
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 21,137,729
Trainable params: 6,423,041
Non-trainable params: 14,714,688
```

TASK 2:

Architecture:
For the Architecture we used the model VGG16 and Alexnet

VGG16:
In this Architecture we used 4 different layers:

1. VGG16 Model Layer
2. Flatten Layer
3. Dense feature Layer
4. Another Dense Layer

Optimizer Task 2: Adam with learning 0.0001

Loss Function Task 2: Binary_Crossentropy

Parameters Task 2: There are a total of 21,138,500 parameters, which includes 6,423,812 trainable and 14,714,688 untrainable ones.

Regularization Task 2: None

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
feature_dense (Dense)        (None, 256)               6422784
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 21,138,500
Trainable params: 6,423,812
Non-trainable params: 14,714,688
_________________________________________________________________
```

Alexnet:

In this Architecture we used 16 different layers:

Optimizer Task 2: Adam with learning 0.0001

Loss Function Task 2: Categorical Cross Entropy

Parameters Task 2: There are a total of 11,325,964 parameters, which includes 11,325,964 trainable and 0 untrainable ones.

Regularization Task 2: Three drop_out layers are used for regularization

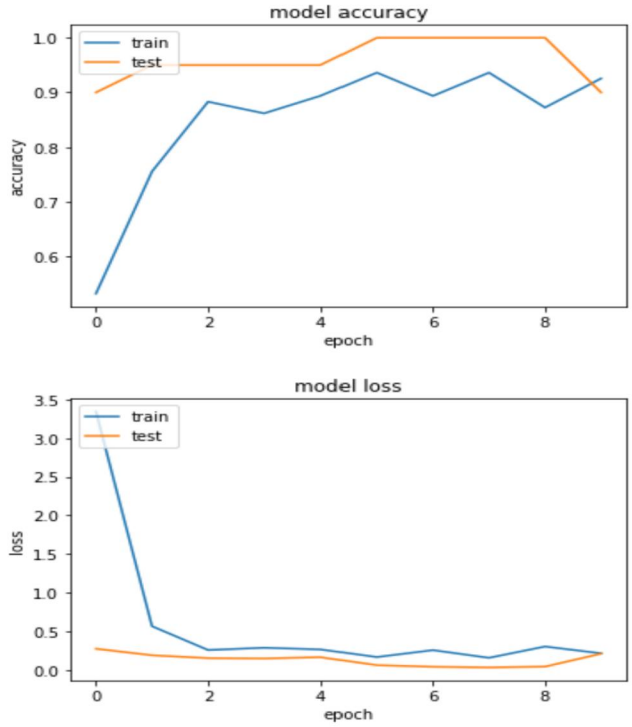
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 54, 54, 96)        34944
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 27, 27, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 17, 17, 256)       2973952
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 6, 384)         885120
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 384)         1327488
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 256)         884992
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
feature_dense (Dense)        (None, 256)               65792
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 4096)              1052672
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 4004
=================================================================
Total params: 11,325,964
Trainable params: 11,325,964
Non-trainable params: 0
_________________________________________________________________
```

Comparison in Task 2:

In Task 2, we used two models, VGG16 and Alexnet.
The VGG16 model had the best results overall over 115 epochs.
VGG16 had the better test accuracy (0.72) and minimal test loss (0.31), while Alexnet had the best training accuracy at 0.83 which is only .3 more than VGG16 (0.8), and the minimal training loss at 0.183.
The reason I say better test results for VGG16 is because it is better than Alexnet at the test accuracy and test loss, and only if off by it a little bit for the training accuracy and training loss. If regularized for both the VGG16 and Alexnet, I think both could be improved potentially, but if both are improved I think VGG16 would still be better than Alexnet because prior to the improvement to both it was still the better of the two. Alexnet however used less parameters than VGG16 and had a lower execution time.

Accuracy and Loss for both Task:

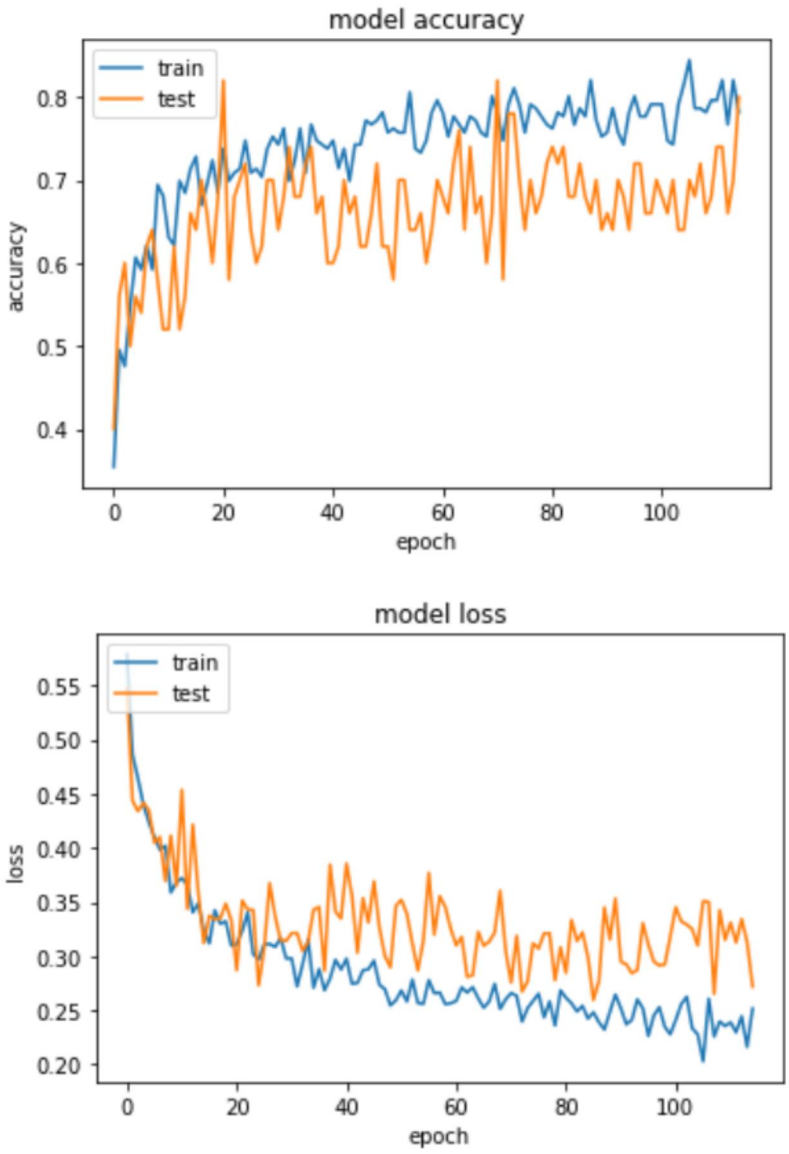
Task 1 (VGG16):



After 40 epoch,

The Training Accuracy = 0.93
The Test Accuracy = 0.81
The Training Loss = 0.174
The Test Loss = 0.27

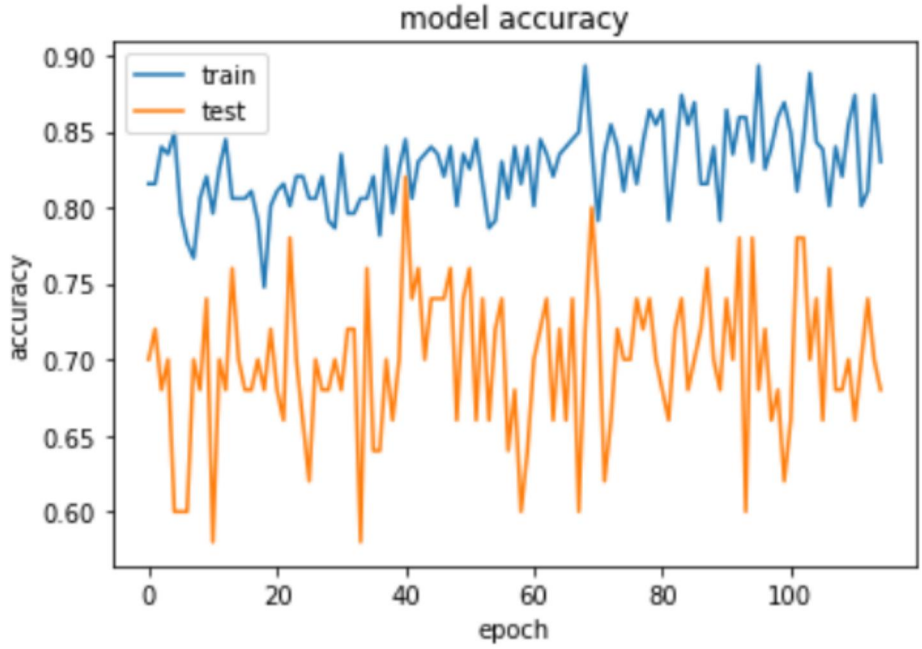
Task 2 (VGG16):

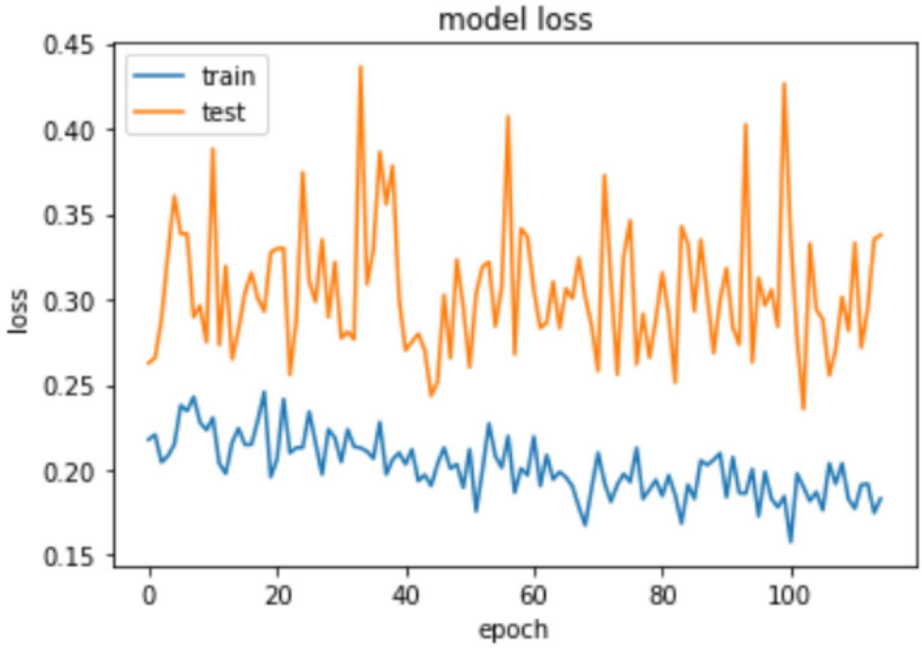


After 115 epoch,

The Training Accuracy = 0.8
The Test Accuracy = 0.72
The Training Loss = 0.25
The Test Loss = 0.31

Task 2 (Alexnet):



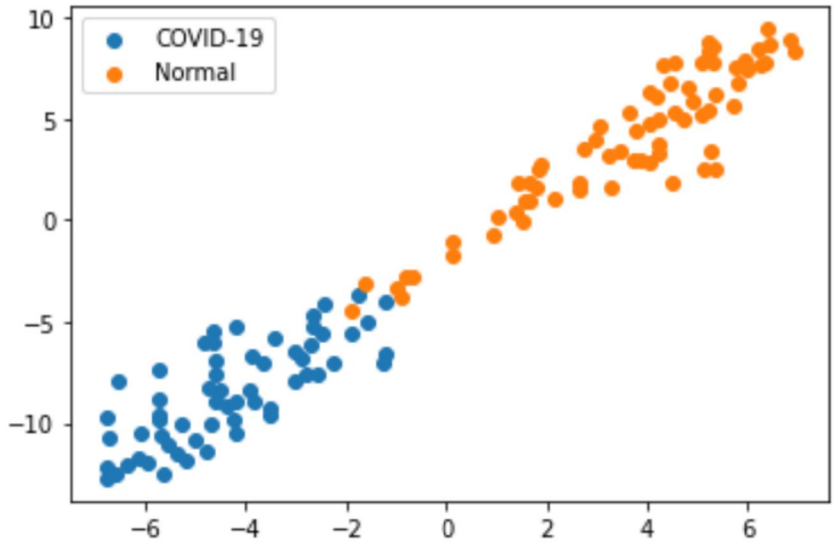


After 115 epoch,

The Training Accuracy = 0.83
The Test Accuracy = 0.66
The Training Loss = 0.1831
The Test Loss = 0.42

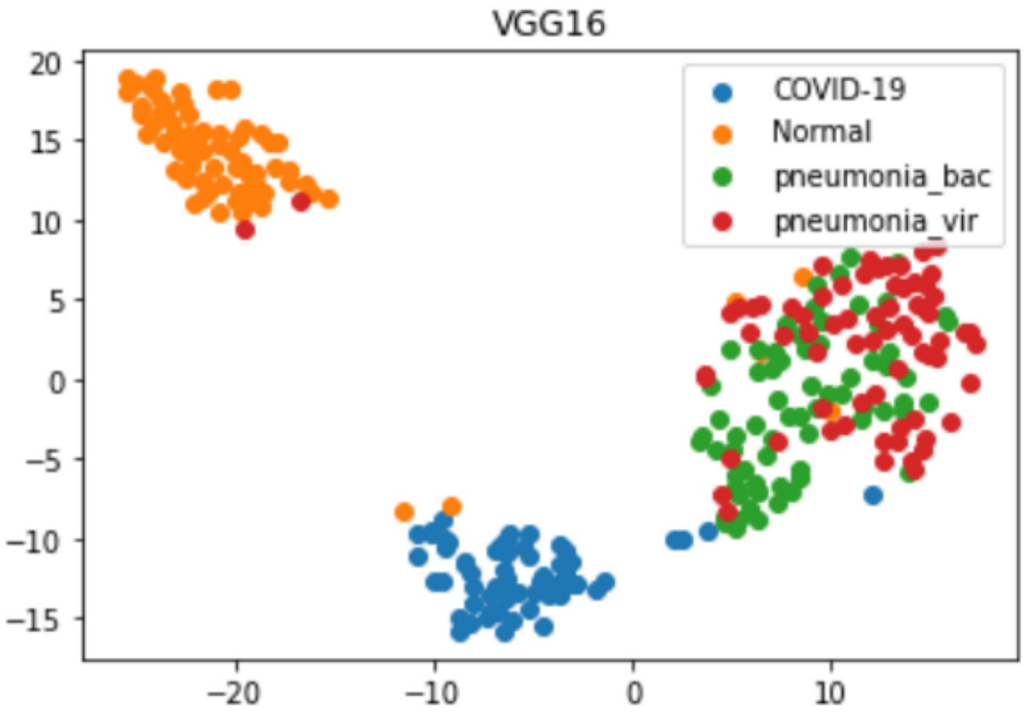
T-SNE Visualization:

Task 1:

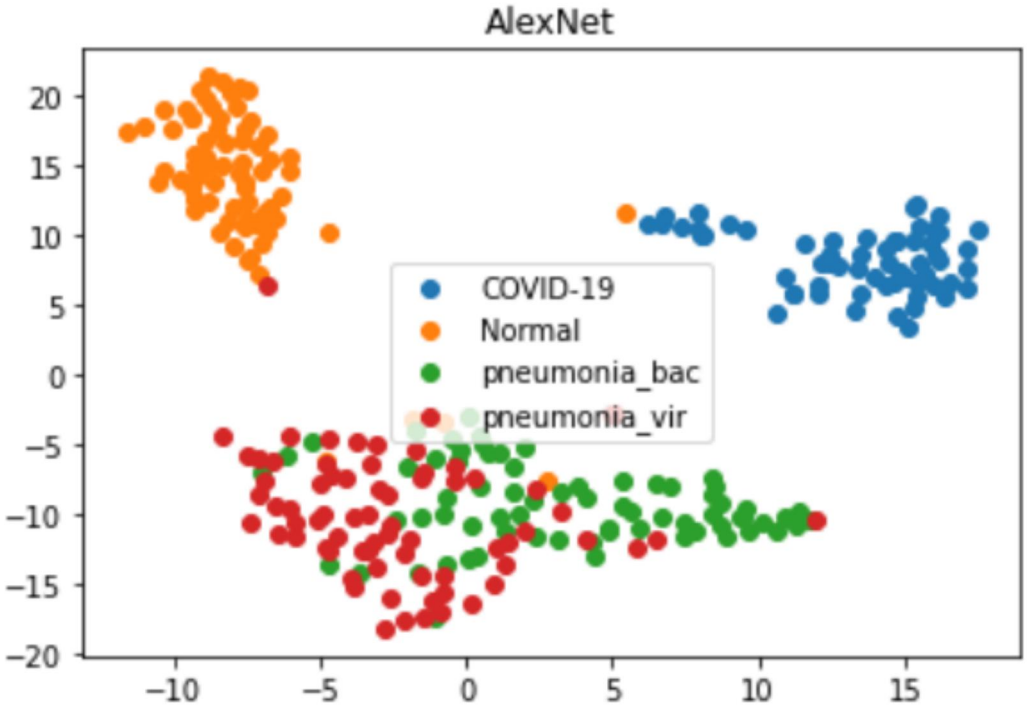


The model VGG16 is really good on the test set because it's able to display the covid 19 and Normal set really well. There are two sets of clusters shown, one in orange and one in blue that shows the overall data.

Task 2:



The model is VGG16 and it performed really well for two of the data sets, for Covid 19 and for normal as we can see the cluster for normal (orange) and the cluster for covid19 (blue). We see both of them clearly, however we can say the same for both the red and the green, the two pneumonia, we can't tell what is what because the data is all mixed up.



The model is Alexnet, we can see that just like VGG16 from above that it performed well for covid 19 (blue), and for normal (orange), however for the two pneumonia, it performed as a good because the cluster of red and green are all mixed up so it’s hard to see.