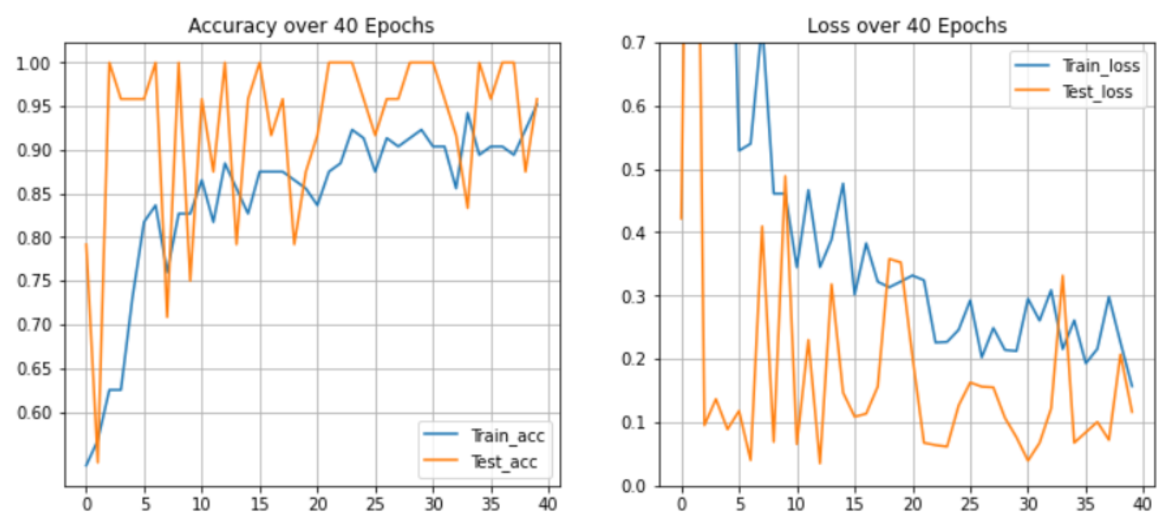
Class Challenge Report
Xinyi Zhao

## Task1:

1. Architectures: The input size of model is (224, 224, 3) which is the image size plus 3. First layer is a VGG16 model layer, which is a pre-trained model, and its weights was obtained by training on imagenet with parameter include_top = false to enable transfer learning. Output size of VGG16 layer is (7, 7, 512).
   Then add the Flatten layer, which flatten the input, and layer dimension is 25088.
   Then add a Dropout layer, and rate is 0.5 to regularizing and avoid overfitting.
   Next, add a Dense layer with dimension = 256.
   Finally, add a Dense layer with dimension = 1 and has activation = 'sigmoid'
   When compile the model:
   Optimizer: SGD optimizer, with learning rate = 0.0005, and momentum = 0.9.
   Loss Function: Binary Crossentropy loss function
   Metrics: accuracy

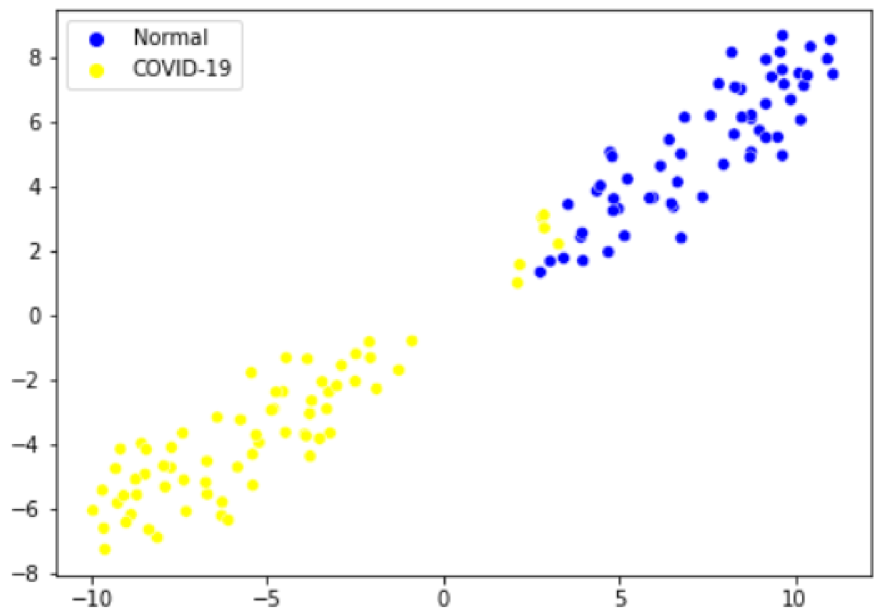
2. Plot and comment on accuracy and loss:




   Train accuracy is high, but is unstable, mostly around 0.75 to 1. Test accuracy is lower than train accuracy, but is growing.
   Train loss is reducing from high, and is unstable, and is mostly around 0.1 to 0.5. Test loss is lower than train loss, and also is unstable, mostly around 0 to 0.3.

3. Plot and comment on t-SNE:

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




The upper cluster is normal which plot as blue, and lower cluster is Covid-19, and plot as yellow.

Most are plotting very well, and clusters are compact, but there still some outliers of Covid-19 points into the normal cluster.

# Task 2:

1. Architectures: The input size of model is (224, 224, 3) which is the image size plus 3. First layer is a VGG16 model layer, which is a pre-trained model, and its weights was obtained by training on imagenet with parameter include_top = false to enable transfer learning. Output size of VGG16 layer is (7, 7, 512).
   Then add a Average Pooling layer, with pooling size = (3,3), so the output dimension is (2, 2, 512).
   Then add the Flatten layer, which flatten the input, and layer dimension is 2048.
   Then add a Dropout layer, and rate is 0.5 to regularizing and avoid overfitting.
   Next, add a Dense layer with dimension = 256, and add a Dense layer with dimension = 4 and has activation = 'softmax'
   When compile the model:
   Optimizer: Adam
   Loss function: Categorical Crossentropy Loss, with parameter from_logits = True
   Metrics: accuracy

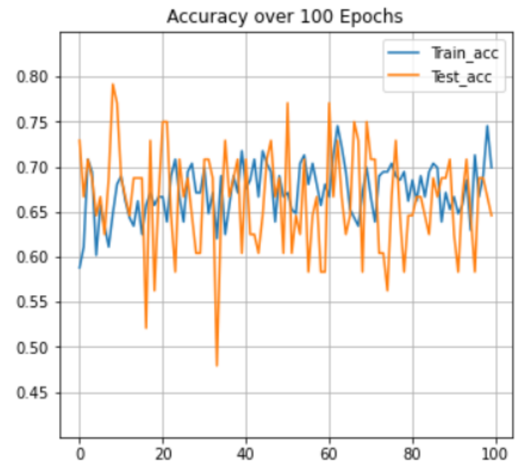
2. Compare performance of different architectures:

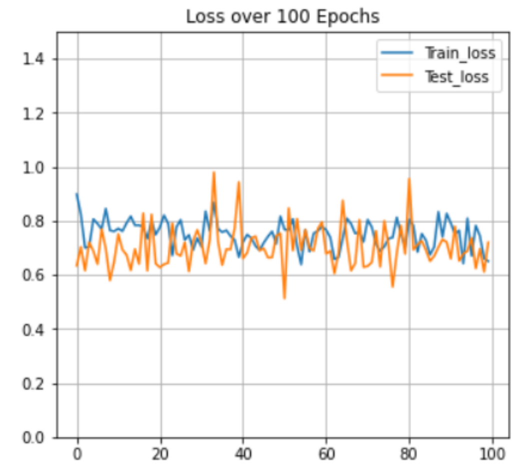
VGG_16 model:

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
average_pooling2d_5 (Average (None, 2, 2, 512)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dropout_10 (Dropout)         (None, 2048)              0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               524544
_________________________________________________________________
dense_10 (Dense)             (None, 4)                 1028
=================================================================
Total params: 15,240,260
Trainable params: 525,572
Non-trainable params: 14,714,688
_________________________________________________________________
```

Architectures: The input size of model is (224, 224, 3) which is the image size plus 3. First layer is a VGG16 model layer, which is a pre-trained model, and its weights was obtained by training on imagenet with parameter include_top = false to enable transfer learning. Output size of VGG16 layer is (7, 7, 512).
Then add a Average Pooling layer, with pooling size = (3,3), so the output dimension is (2, 2, 512).
Then add the Flatten layer, which flatten the input, and layer dimension is 2048.
Then add a Dropout layer, and rate is 0.5 to regularizing and avoid overfitting.
Next, add a Dense layer with dimension = 256.
Finally add a Dense layer with dimension = 4 and has activation = ‘softmax’

Total number of parameters in the final is 15,240,260; Trainable parameters is 525,572; Non-trainable parameters is 14,714,688.

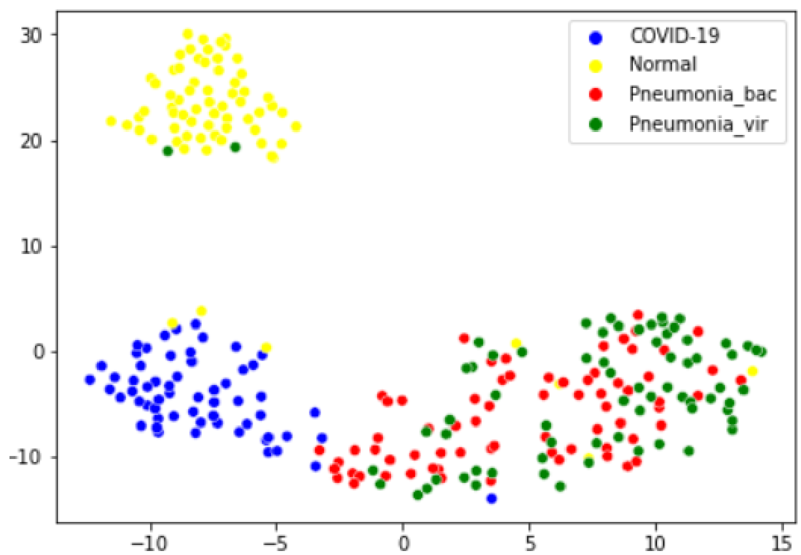





```
x6 = model6.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
                               use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x6[0])
print('Test accuracy:',x6[1])
```

```
36/36 [==============================] - 3s 95ms/step - loss: 0.6917 - acc: 0.7222
Test loss: 0.6917243599891663
Test accuracy: 0.7222222089767456
```

Test accuracy = 0.7222
Test loss = 0.6917

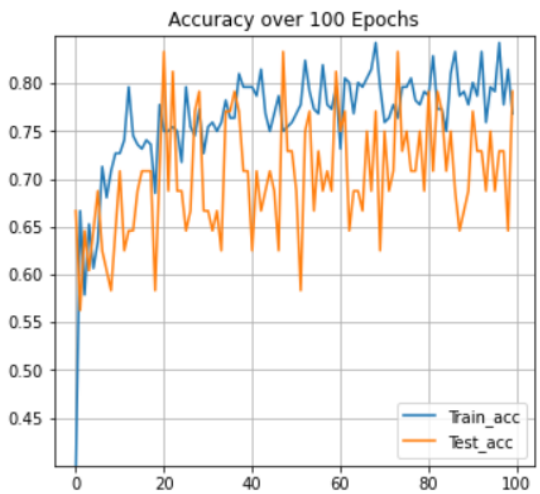


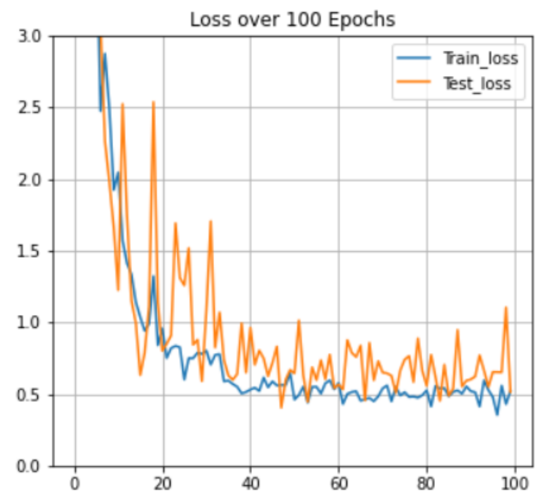
Mobile Net model:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Function (None, 7, 7, 1024)        3228864
_________________________________________________________________
average_pooling2d_1 (Average (None, 2, 2, 1024)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               1048832
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 4,278,724
Trainable params: 1,049,860
Non-trainable params: 3,228,864
_________________________________________________________________
```

Architectures: The input size of model is (224, 224, 3) which is the image size plus 3.
First layer is a mobile net model layer, which is a pre-trained model, and its weights was obtained by training on imagenet with parameter include_top = false to enable transfer learning. Output size of MobileNet layer is (7, 7, 1024).
Then add a Average Pooling layer, with pooling size = (3,3), so the output dimension is (2, 2, 1024).
Then add the Flatten layer, which flatten the input, and layer dimension is 4096.
Then add a Dropout layer, and rate is 0.5 to regularizing and avoid overfitting.
Next, add a Dense layer with dimension = 256.
Finally add a Dense layer with dimension = 4 and has activation = 'softmax'

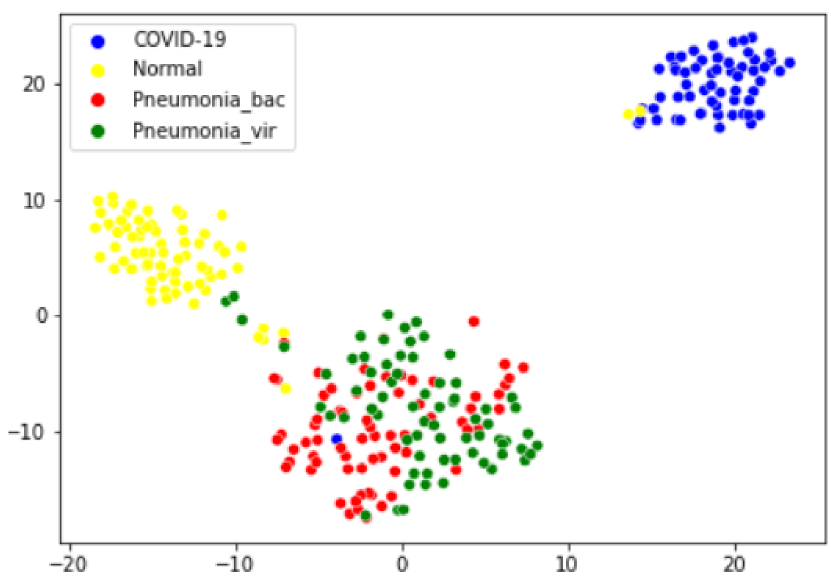
Total number of parameters in the final is 4,278,724; Trainable parameters is 1,049,860; Non-trainable parameters is 3,228,864.
Total number of parameters in the final is much less than the model uses vgg16 as pre-trained model. Trainable parameters is much more than the vgg16 model; Non-trainable parameters is less than vgg 16 model.




```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 2s 51ms/step - loss: 0.8847 - acc: 0.5556
Test loss: 0.8847371935844421
Test accuracy: 0.5555555820465088
```

Test accuracy = 0.5556, which is much lower than model with vgg16 model as pre-trained model
Test loss = 0.8847, which is higher than model with vgg16 model as pre-trained model

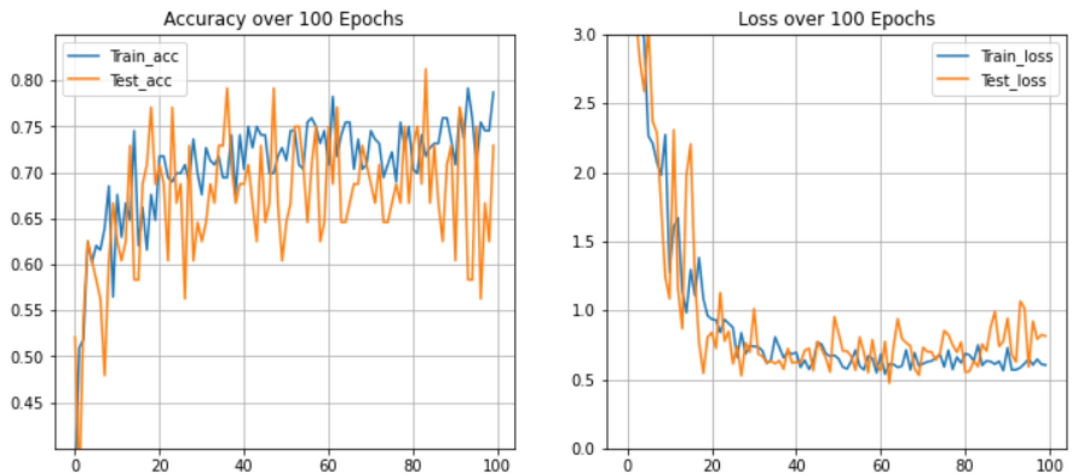


DenseNet121 model:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Functional)     (None, 7, 7, 1024)        7037504
_________________________________________________________________
average_pooling2d (AveragePo (None, 2, 2, 1024)        0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               1048832
_________________________________________________________________
dense (Dense)                (None, 4)                 1028
=================================================================
Total params: 8,087,364
Trainable params: 1,049,860
Non-trainable params: 7,037,504
_________________________________________________________________
```

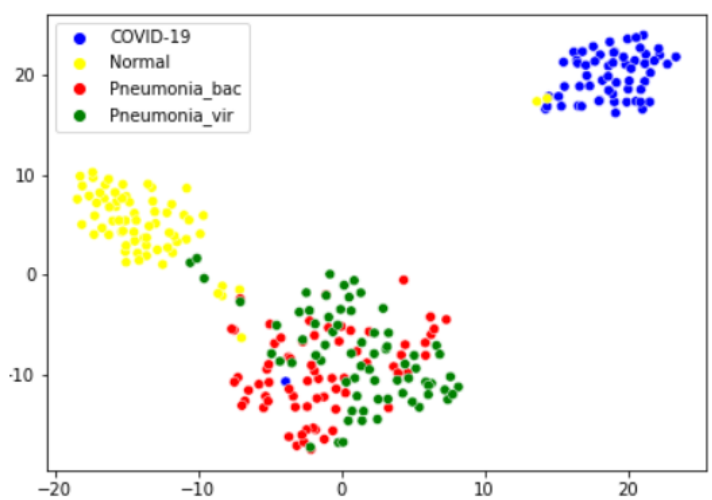
Architectures: The input size of model is (224, 224, 3) which is the image size plus 3. First layer is a DenseNet 121 model layer, which is a pre-trained model, and its weights was obtained by training on imagenet with parameter include_top = false to enable transfer learning. Output size of DenseNet 121layer is (7, 7, 1024).
Then add a Average Pooling layer, with pooling size = (3,3), so the output dimension is (2, 2, 1024).
Then add the Flatten layer, which flatten the input, and layer dimension is 4096.
Then add a Dropout layer, and rate is 0.5 to regularizing and avoid overfitting.
Next, add a Dense layer with dimension = 256.
Finally add a Dense layer with dimension = 4 and has activation = ‘softmax’

Total number of parameters in the final is 8,087,364; Trainable parameters is 1,049,860; Non-trainable parameters is 7,037,504.
Total number of parameters in the final is much less than the model uses vgg16 as pre-trained model, and is more than mobileNet as pre-trained model. Trainable parameters is much more than the vgg16 model, and is same as mobileNet model; Non-trainable parameters is less than vgg 16 model, and is more than mobileNet model.

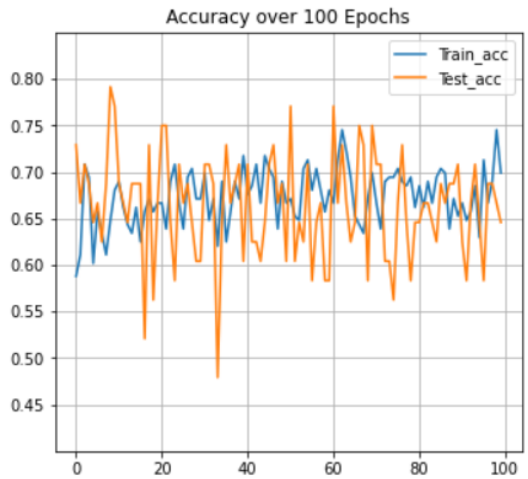


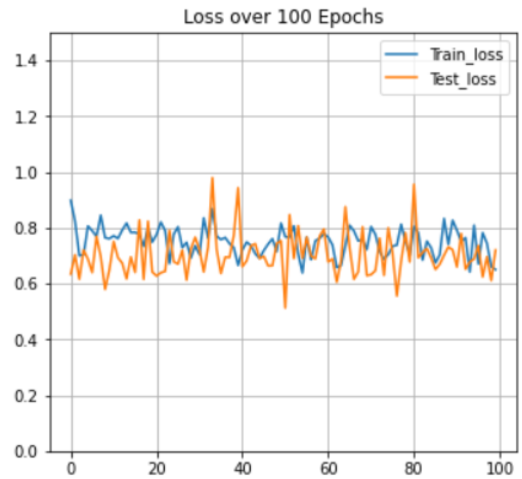

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 3s 74ms/step - loss: 0.7143 - acc: 0.6667
Test loss: 0.7143376469612122
Test accuracy: 0.6666666865348816
```

Test accuracy = 0.6667, which is lower than model with vgg16 model as pre-trained model, but higher than mobileNet model as pre-trained model
Test loss = 0.7143, which is higher than model with vgg16 model as pre-trained model, and is lower than mobileNet model



3. Plot and comment on accuracy and loss:



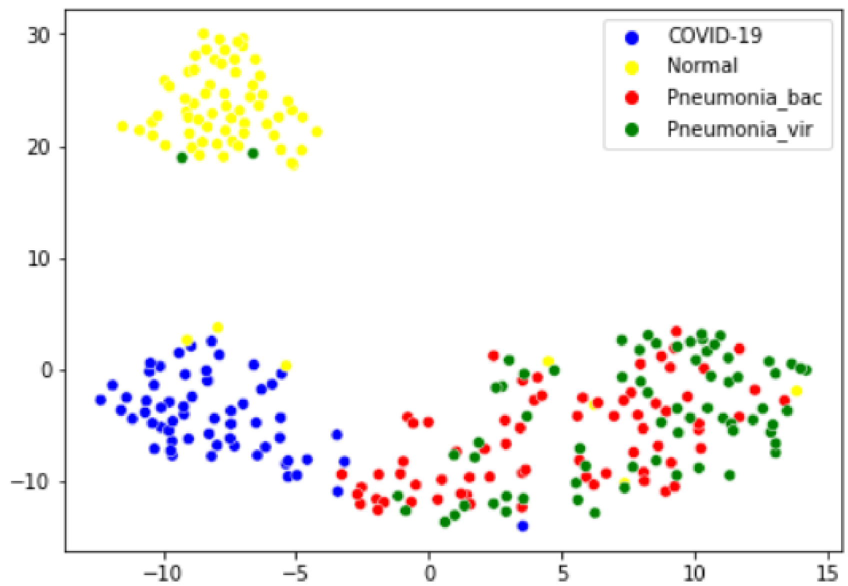




```
x6 = model6.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
                               use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x6[0])
print('Test accuracy:',x6[1])
```

```
36/36 [==============================] - 3s 95ms/step - loss: 0.6917 - acc: 0.7222
Test loss: 0.6917243599891663
Test accuracy: 0.7222222089767456
```

Test and train accuracy are unstable in plot. Test accuracy is lower than train accuracy, and the result of test accuracy is very high and is about 0.7222.
Test loss is about 0.6917, and train loss is lower than test loss.

4. Plot and comment on t-SNE:




The upper yellow cluster is Normal, lower left blue cluster is COVID-19, lower right red is Pneumonia_bac, and lower right green is Pneumonia_vir.
Most are plotting very well, and clusters are compact.
For cluster COVID-19 and cluster Normal, the accuracy is very high and the plot has clear boundaries from others, except some little outliers.
For cluster Pneumonia_bac and cluster Pneumonia_vir, although the boundary is not distinct, most Pneumonia_bac points are at the left side, and most Pneumonia_vir are at the right side. There are also some outliers from COVID-19 and Normal points.

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

# Task 1 [Total points: 30]

## Setup

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

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

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

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

## Data

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

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

|--all
|--------train
|--------test
|--two
|--------train
|--------test

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

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

In [2]:
```
import os

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

```
os.environ['OMP_NUM_THREADS'] = '1'
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
tf.__version__
```

Out[2]: '2.4.1'

### Load Image Data

In [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    = 8  # try reducing batch size or freeze more layers if your GPU runs out
NUM_EPOCHS    = 40
LEARNING_RATE = 0.0007 # start off with high rate first 0.001 and experiment with reduc
```

### Generate Training and Validation Batches

In [4]:

```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center
                                   featurewise_std_normalization = True,width_shift_ran
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0
                                   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")
```

```
D:\downloads\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.
py:342: UserWarning: This ImageDataGenerator specifies `zca_whitening` which overrides s
etting of`featurewise_std_normalization`.
  warnings.warn('This ImageDataGenerator specifies '
Found 104 images belonging to 2 classes.
Found 26 images belonging to 2 classes.
```

### [10 points] Build Model

Hint: Starting from a pre-trained model typically helps performance on a new task, e.g. starting with weights obtained by training on ImageNet.

In [8]:

```
#raise NotImplementedError("Build your model based on an architecture of your choice "
#                          "A sample model summary is shown below")

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

vgg_model.trainable = False
```

```
model = tf.keras.models.Sequential()
model.add(vgg_model)
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dropout(rate = 0.5))
model.add(tf.keras.layers.Dense(256, name = "dense_feature"))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688  
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dropout_1 (Dropout)          (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 [9]:

```
model.compile(optimizer=tf.keras.optimizers.SGD(lr = LEARNING_RATE, momentum = 0.9),
              loss='binary_crossentropy',
              metrics=['acc'])
#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

#raise NotImplementedError("Use the model.fit function to train your network")

history = model.fit(
    train_batches,
    epochs=NUM_EPOCHS, steps_per_epoch=STEP_SIZE_TRAIN,
    batch_size=BATCH_SIZE,
    validation_data = valid_batches, validation_steps=STEP_SIZE_VALID
)
```

```
13
4
Epoch 1/40
13/13 [==============================] - 14s 1s/step - loss: 1.0473 - acc: 0.5205 - val_
loss: 0.4219 - val_acc: 0.7917
Epoch 2/40
13/13 [==============================] - 14s 1s/step - loss: 1.4690 - acc: 0.6197 - val_
loss: 1.7732 - val_acc: 0.5417
Epoch 3/40
13/13 [==============================] - 14s 1s/step - loss: 2.3286 - acc: 0.6026 - val_
loss: 0.0952 - val_acc: 1.0000
Epoch 4/40
13/13 [==============================] - 14s 1s/step - loss: 1.3947 - acc: 0.6213 - val_
loss: 0.1366 - val_acc: 0.9583
Epoch 5/40
```

```
13/13 [==============================] - 14s 1s/step - loss: 1.2291 - acc: 0.7435 - val_
loss: 0.0889 - val_acc: 0.9583
Epoch 6/40
13/13 [==============================] - 14s 1s/step - loss: 0.4377 - acc: 0.8310 - val_
loss: 0.1180 - val_acc: 0.9583
Epoch 7/40
13/13 [==============================] - 13s 1s/step - loss: 0.3942 - acc: 0.8828 - val_
loss: 0.0402 - val_acc: 1.0000
Epoch 8/40
13/13 [==============================] - 14s 1s/step - loss: 0.6177 - acc: 0.8061 - val_
loss: 0.4096 - val_acc: 0.7083
Epoch 9/40
13/13 [==============================] - 14s 1s/step - loss: 0.5702 - acc: 0.8180 - val_
loss: 0.0691 - val_acc: 1.0000
Epoch 10/40
13/13 [==============================] - 14s 1s/step - loss: 0.5058 - acc: 0.8330 - val_
loss: 0.4889 - val_acc: 0.7500
Epoch 11/40
13/13 [==============================] - 14s 1s/step - loss: 0.4460 - acc: 0.8269 - val_
loss: 0.0654 - val_acc: 0.9583
Epoch 12/40
13/13 [==============================] - 15s 1s/step - loss: 0.4639 - acc: 0.8336 - val_
loss: 0.2301 - val_acc: 0.8750
Epoch 13/40
13/13 [==============================] - 14s 1s/step - loss: 0.6016 - acc: 0.8109 - val_
loss: 0.0350 - val_acc: 1.0000
Epoch 14/40
13/13 [==============================] - 14s 1s/step - loss: 0.2560 - acc: 0.8896 - val_
loss: 0.3182 - val_acc: 0.7917
Epoch 15/40
13/13 [==============================] - 14s 1s/step - loss: 0.4859 - acc: 0.8277 - val_
loss: 0.1470 - val_acc: 0.9583
Epoch 16/40
13/13 [==============================] - 13s 1s/step - loss: 0.3424 - acc: 0.8780 - val_
loss: 0.1088 - val_acc: 1.0000
Epoch 17/40
13/13 [==============================] - 14s 1s/step - loss: 0.2955 - acc: 0.9111 - val_
loss: 0.1135 - val_acc: 0.9167
Epoch 18/40
13/13 [==============================] - 14s 1s/step - loss: 0.3895 - acc: 0.8698 - val_
loss: 0.1564 - val_acc: 0.9583
Epoch 19/40
13/13 [==============================] - 14s 1s/step - loss: 0.2640 - acc: 0.8810 - val_
loss: 0.3580 - val_acc: 0.7917
Epoch 20/40
13/13 [==============================] - 14s 1s/step - loss: 0.2908 - acc: 0.8599 - val_
loss: 0.3525 - val_acc: 0.8750
Epoch 21/40
13/13 [==============================] - 14s 1s/step - loss: 0.3536 - acc: 0.8377 - val_
loss: 0.2041 - val_acc: 0.9167
Epoch 22/40
13/13 [==============================] - 14s 1s/step - loss: 0.2776 - acc: 0.8982 - val_
loss: 0.0675 - val_acc: 1.0000
Epoch 23/40
13/13 [==============================] - 14s 1s/step - loss: 0.2683 - acc: 0.8600 - val_
loss: 0.0639 - val_acc: 1.0000
Epoch 24/40
13/13 [==============================] - 14s 1s/step - loss: 0.1857 - acc: 0.9268 - val_
loss: 0.0616 - val_acc: 1.0000
Epoch 25/40
13/13 [==============================] - 14s 1s/step - loss: 0.2305 - acc: 0.9152 - val_
loss: 0.1266 - val_acc: 0.9583
Epoch 26/40
13/13 [==============================] - 14s 1s/step - loss: 0.3050 - acc: 0.8771 - val_
loss: 0.1628 - val_acc: 0.9167
```

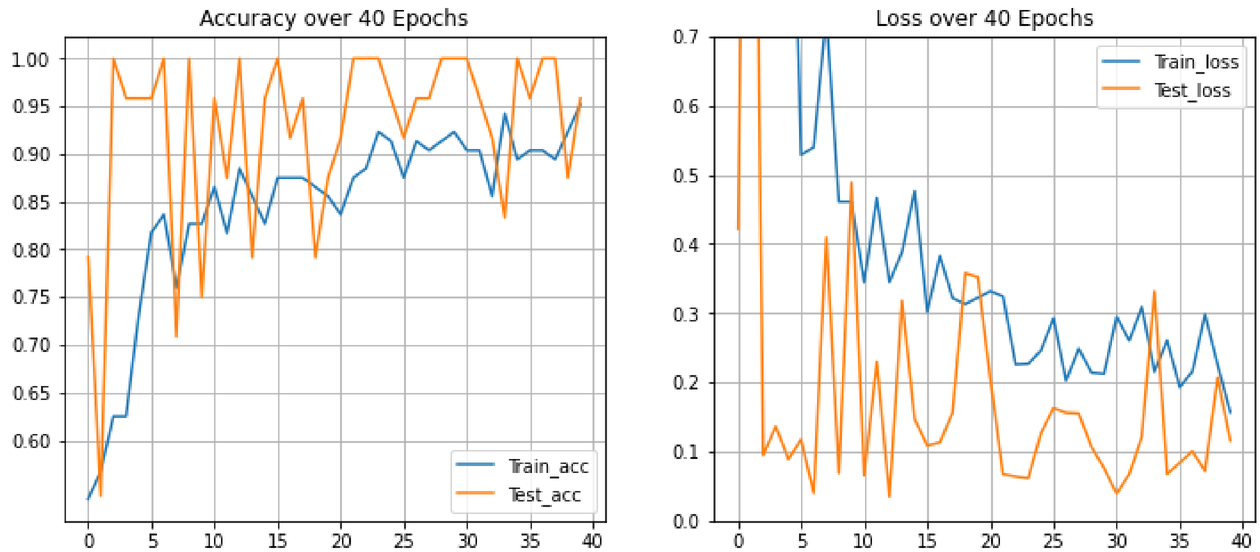
```
Epoch 27/40
13/13 [==============================] - 14s 1s/step - loss: 0.1900 - acc: 0.9157 - val_
loss: 0.1563 - val_acc: 0.9583
Epoch 28/40
13/13 [==============================] - 14s 1s/step - loss: 0.2416 - acc: 0.8812 - val_
loss: 0.1552 - val_acc: 0.9583
Epoch 29/40
13/13 [==============================] - 14s 1s/step - loss: 0.2312 - acc: 0.9030 - val_
loss: 0.1071 - val_acc: 1.0000
Epoch 30/40
13/13 [==============================] - 14s 1s/step - loss: 0.2250 - acc: 0.9073 - val_
loss: 0.0763 - val_acc: 1.0000
Epoch 31/40
13/13 [==============================] - 14s 1s/step - loss: 0.2804 - acc: 0.8635 - val_
loss: 0.0394 - val_acc: 1.0000
Epoch 32/40
13/13 [==============================] - 14s 1s/step - loss: 0.2886 - acc: 0.9203 - val_
loss: 0.0674 - val_acc: 0.9583
Epoch 33/40
13/13 [==============================] - 14s 1s/step - loss: 0.2913 - acc: 0.8499 - val_
loss: 0.1217 - val_acc: 0.9167
Epoch 34/40
13/13 [==============================] - 14s 1s/step - loss: 0.3020 - acc: 0.9070 - val_
loss: 0.3317 - val_acc: 0.8333
Epoch 35/40
13/13 [==============================] - 14s 1s/step - loss: 0.2947 - acc: 0.8822 - val_
loss: 0.0676 - val_acc: 1.0000
Epoch 36/40
13/13 [==============================] - 14s 1s/step - loss: 0.1580 - acc: 0.9355 - val_
loss: 0.0841 - val_acc: 0.9583
Epoch 37/40
13/13 [==============================] - 14s 1s/step - loss: 0.2217 - acc: 0.9146 - val_
loss: 0.1006 - val_acc: 1.0000
Epoch 38/40
13/13 [==============================] - 14s 1s/step - loss: 0.3761 - acc: 0.8788 - val_
loss: 0.0721 - val_acc: 1.0000
Epoch 39/40
13/13 [==============================] - 14s 1s/step - loss: 0.2262 - acc: 0.9176 - val_
loss: 0.2073 - val_acc: 0.8750
Epoch 40/40
13/13 [==============================] - 14s 1s/step - loss: 0.2592 - acc: 0.9252 - val_
loss: 0.1164 - val_acc: 0.9583
```

## [5 points] Plot Accuracy and Loss During Training

In [11]:

```python
import matplotlib.pyplot as plt

#raise NotImplementedError("Plot the accuracy and the loss during training")
plt.rcParams['figure.figsize'] = [12, 5]
fig, (ax1, ax2) = plt.subplots(1, 2)


ax1.plot(history.history['acc'], label = 'Train_acc')
ax1.plot(history.history['val_acc'], label = 'Test_acc')
ax1.set_title("Accuracy over 40 Epochs")
ax1.legend(['Train_acc','Test_acc'])
ax1.grid()
ax1.set_yticks([0.6, 0.65,0.7,0.75,0.8,0.85,0.9,0.95,1])

ax2.plot(history.history['loss'], label = 'Train_loss')
ax2.plot(history.history['val_loss'], label = 'Test_loss')
ax2.set_title("Loss over 40 Epochs")
ax2.legend(['Train_loss','Test_loss'])
```

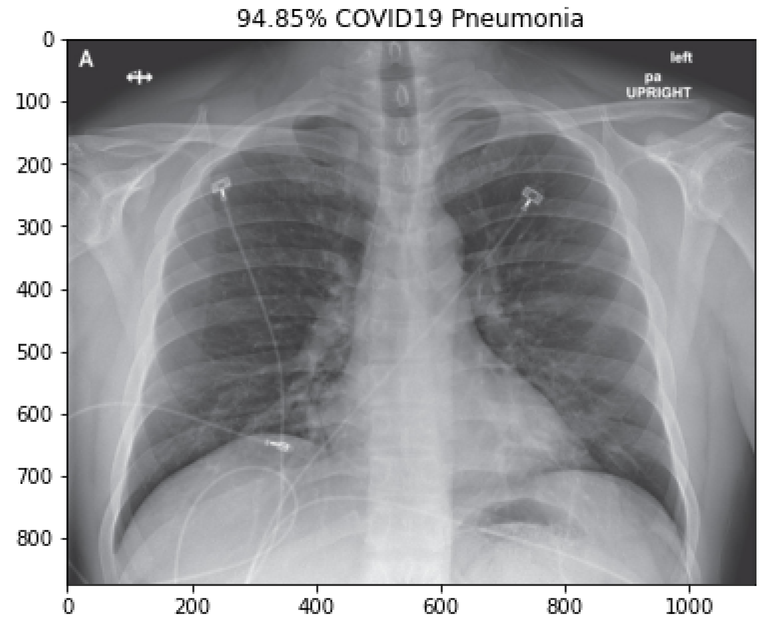
```
ax2.set_ylim([0,0.7])
ax2.grid()


plt.show()
```



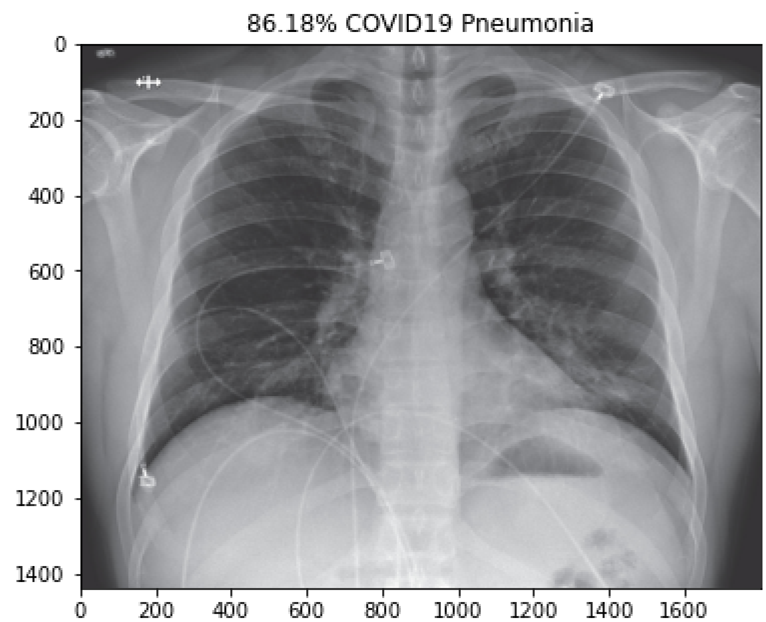

## Plot Test Results

In [12]:
```
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,cla
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()
```

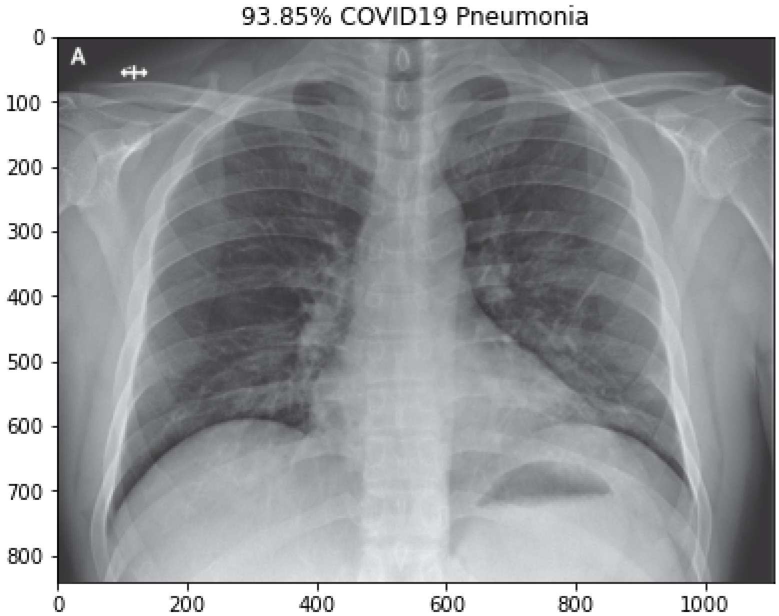
```
Found 18 images belonging to 2 classes.
D:\downloads\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:190
5: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future
version. Please use `Model.predict`, which supports generators.
  warnings.warn('`Model.predict_generator` is deprecated and '
18/18 [==============================] - 3s 136ms/step
covid\nejmoa2001191_f3-PA.jpeg
```



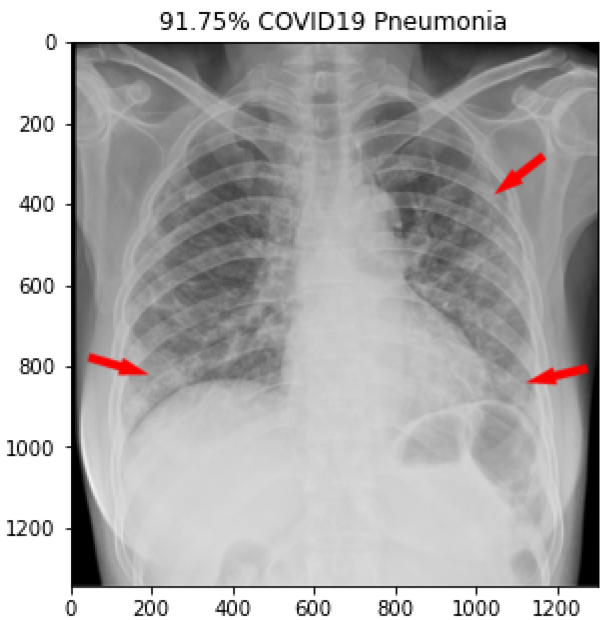

covid\nejmoa2001191_f4.jpeg



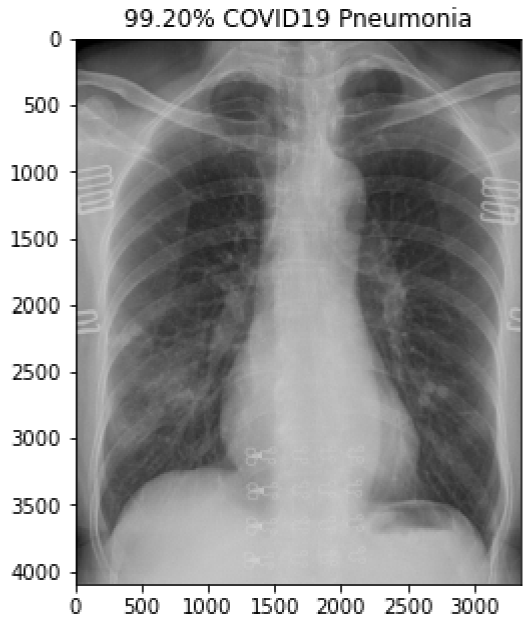

covid\nejmoa2001191_f5-PA.jpeg



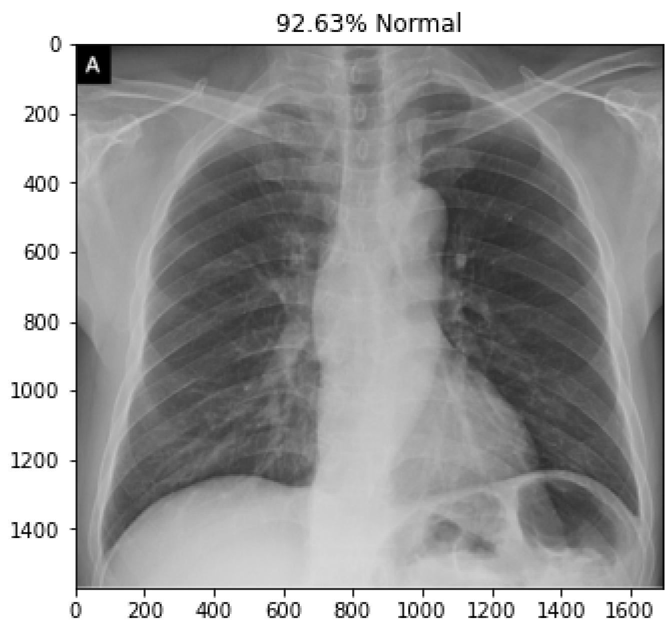

covid\radiol.2020200490.fig3.jpeg



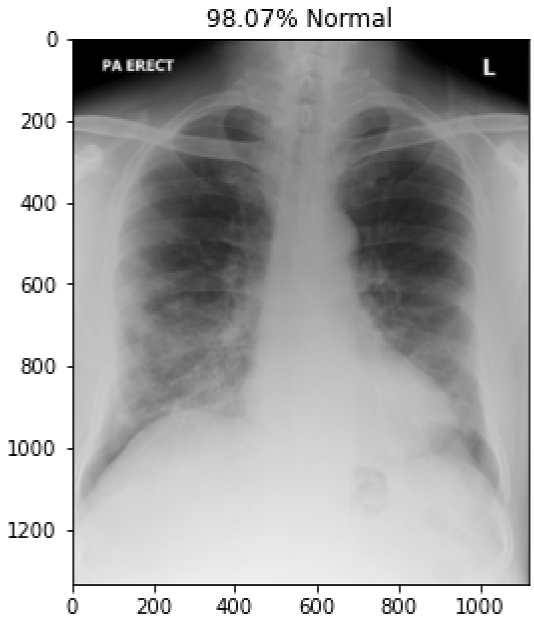

covid\ryct.2020200028.fig1a.jpeg



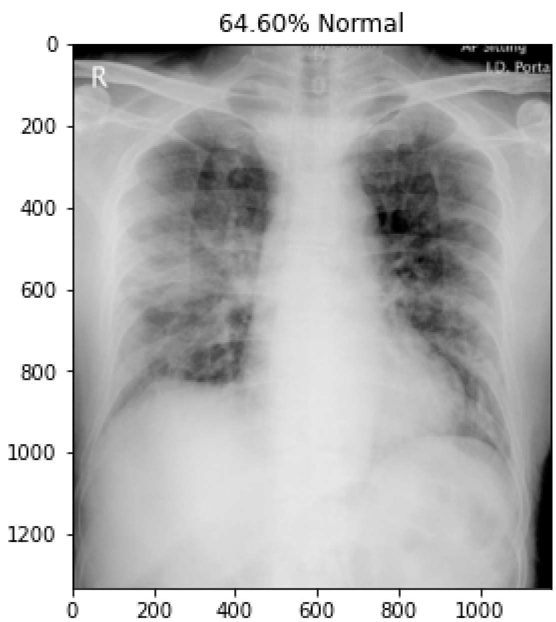

covid\ryct.2020200034.fig2.jpeg



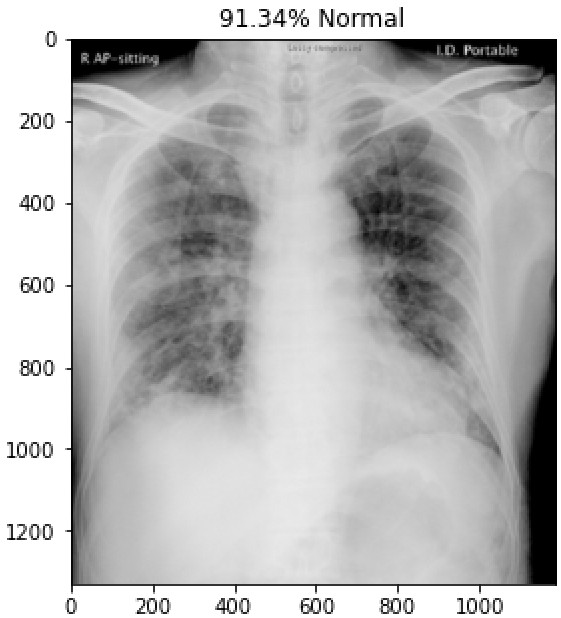

covid\ryct.2020200034.fig5-day0.jpeg



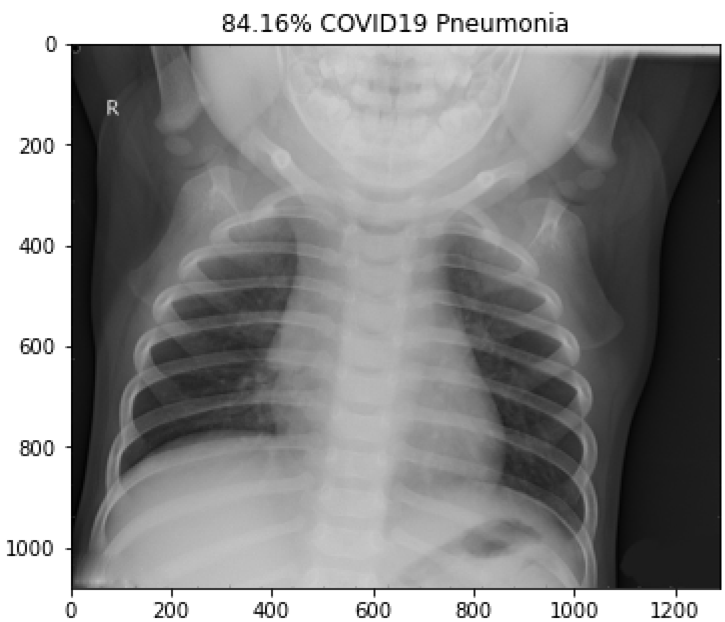

covid\ryct.2020200034.fig5-day4.jpeg



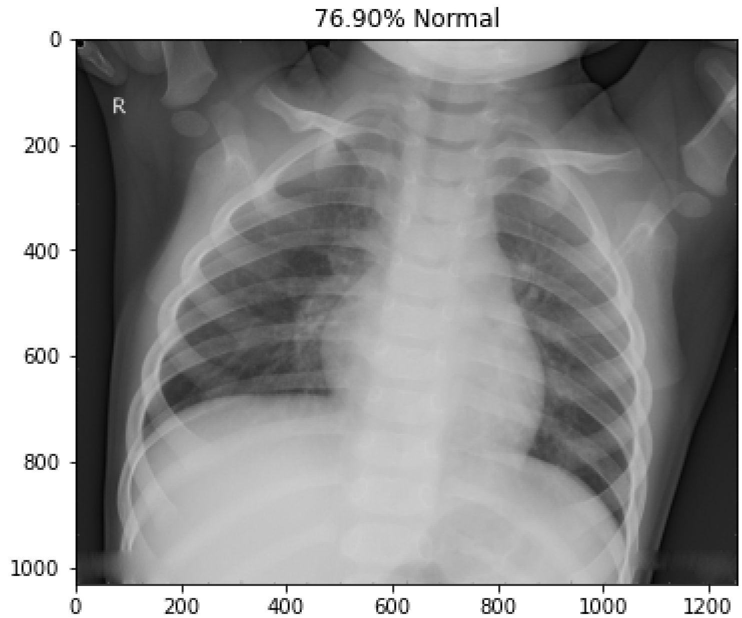

covid\ryct.2020200034.fig5-day7.jpeg



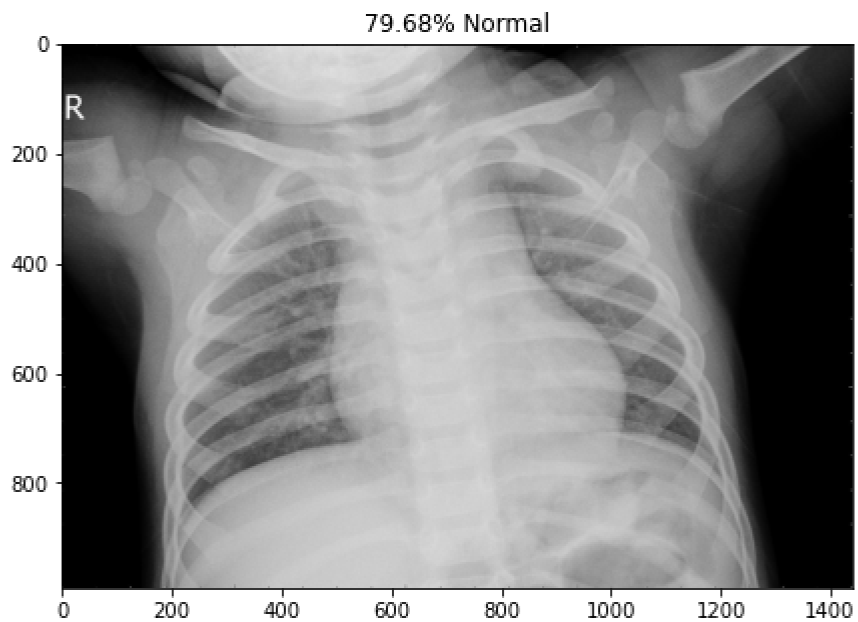

normal\NORMAL2-IM-1385-0001.jpeg



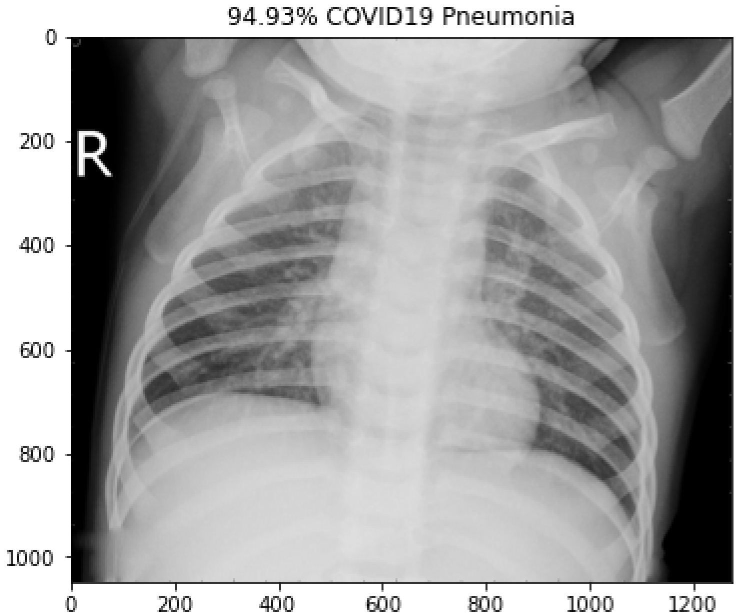

normal\NORMAL2-IM-1396-0001.jpeg



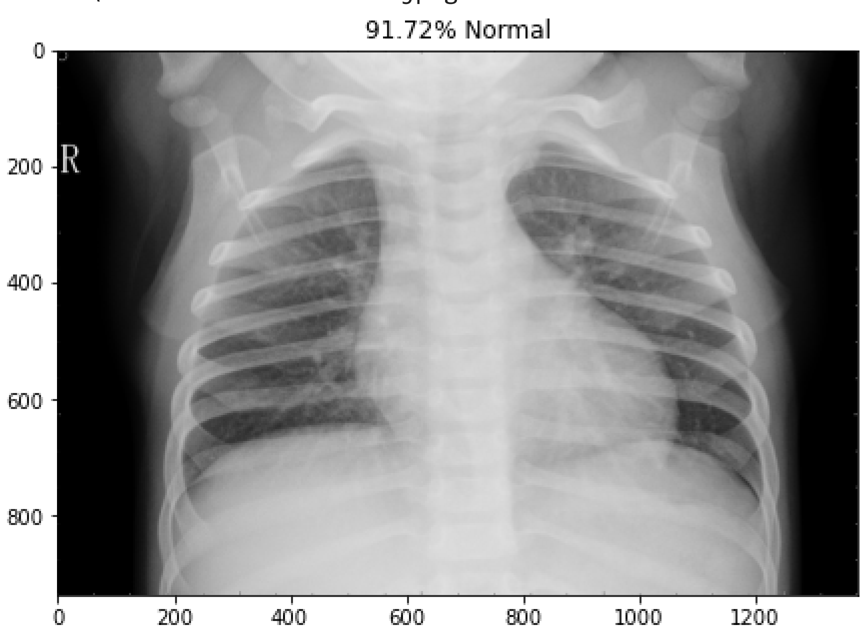

normal\NORMAL2-IM-1400-0001.jpeg



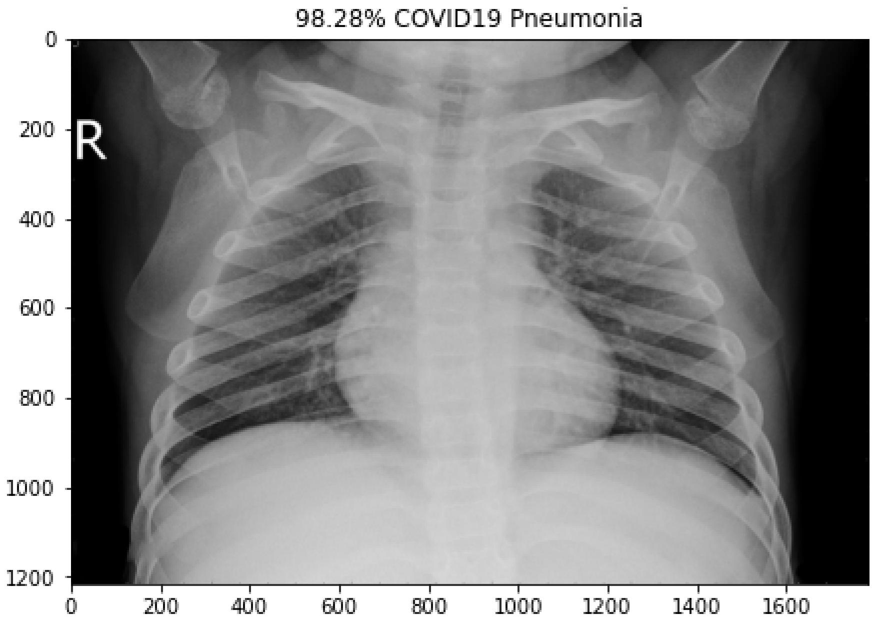

normal\NORMAL2-IM-1401-0001.jpeg



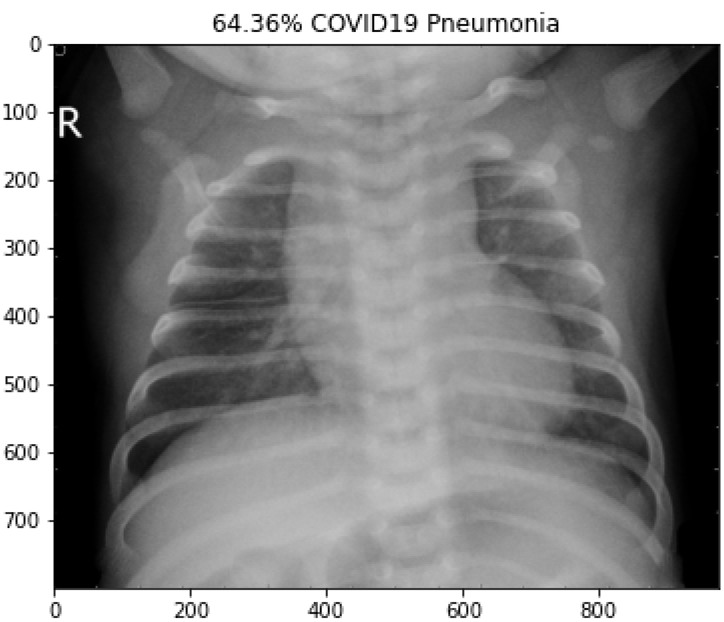

normal\NORMAL2-IM-1406-0001.jpeg



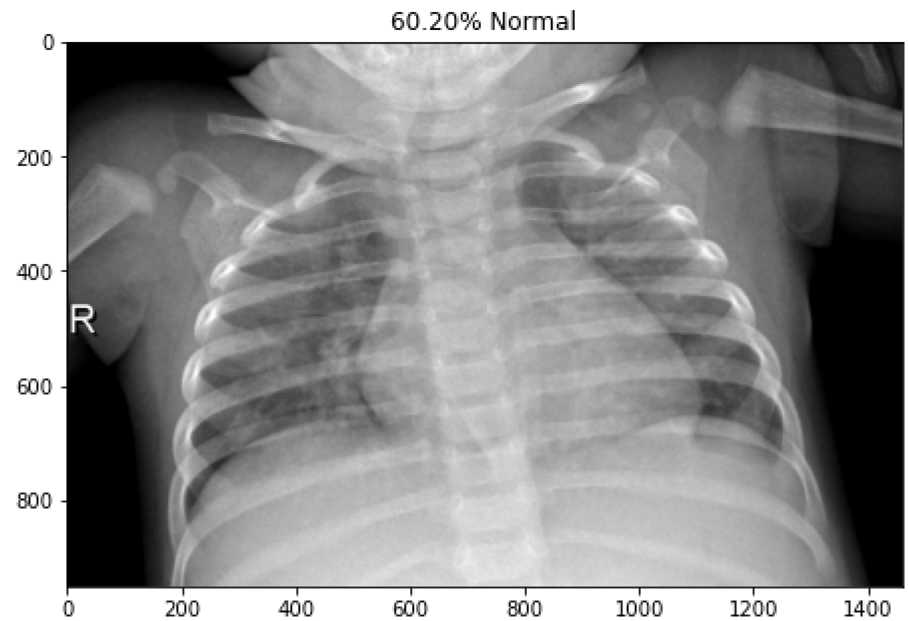

normal\NORMAL2-IM-1412-0001.jpeg



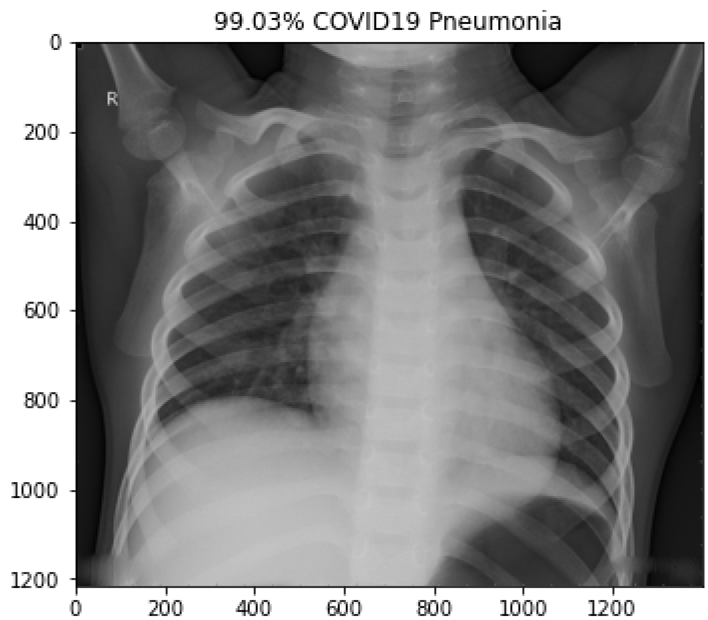

```
normal\NORMAL2-IM-1419-0001.jpeg
```




```
normal\NORMAL2-IM-1422-0001.jpeg
```




normal\NORMAL2-IM-1423-0001.jpeg



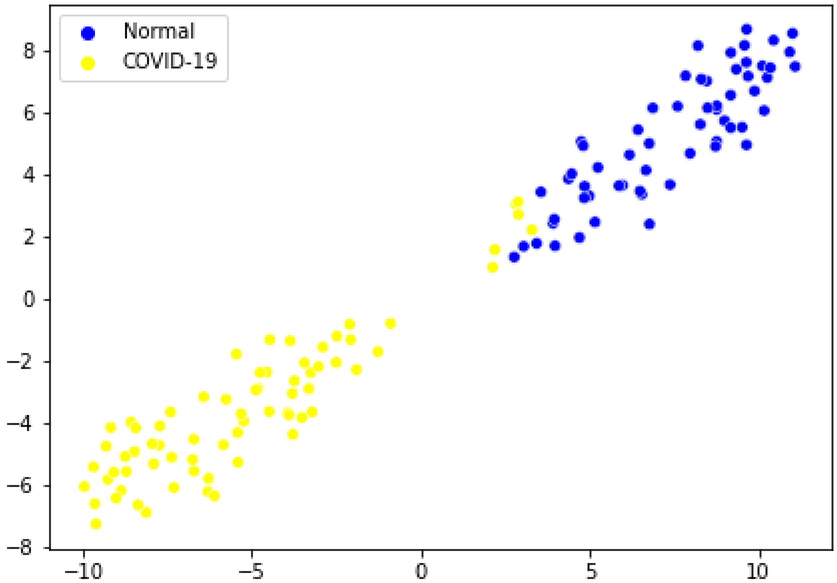

# [10 points] TSNE Plot

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 [31]:
```
from sklearn.manifold import TSNE
import seaborn as sns

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_S
                                                       batch_size=1,shuffle=False,seed=42,cl

#raise NotImplementedError("Extract features from the tsne_data_generator and fit a t-S
```

```
#                          "and plot the resulting 2D features of the two classes.")

plt.rcParams['figure.figsize'] = [7, 5]

intermediate_output = intermediate_layer_model.predict(tsne_data_generator)
tsne = TSNE(n_components=2)
results = tsne.fit_transform(intermediate_output)
h = tsne_data_generator.labels
sns.scatterplot(x = results[:,0], y = results[:,1], legend = 'full', hue= h, palette=["


L=plt.legend()
L.get_texts()[0].set_text('Normal')
L.get_texts()[1].set_text('COVID-19')
```

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



# 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 [15]:
```
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[15]: '2.4.1'

### Load Image Data

In [16]:

```
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    = 8  # try reducing batch size or freeze more layers if your GPU runs out
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0005 # start off with high rate first 0.001 and experiment with reduc
```

### Generate Training and Validation Batches

In [17]:

```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center
                                   featurewise_std_normalization = True,width_shift_ran
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0
                                   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.
```

### [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 [18]:

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

vgg_model.trainable = False

model6 = tf.keras.models.Sequential()
model6.add(vgg_model)
model6.add(tf.keras.layers.AveragePooling2D(pool_size=(3,3)))
model6.add(tf.keras.layers.Flatten())
model6.add(tf.keras.layers.Dropout(rate = 0.5))
model6.add(tf.keras.layers.Dense(256, name = "dense_feature"))
```

```
model6.add(tf.keras.layers.Dense(4, activation='softmax'))
model6.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 7, 7, 512)         14714688
_________________________________________________________________
average_pooling2d_1 (Average (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               524544
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 15,240,260
Trainable params: 525,572
Non-trainable params: 14,714,688
_________________________________________________________________
```

## [5 points] Train Model

In [29]:

```
model6.compile(optimizer='adam',
                loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
                metrics=['acc'])

#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

#raise NotImplementedError("Use the model.fit function to train your network")

history = model6.fit(
    train_batches,
    epochs=NUM_EPOCHS, steps_per_epoch=STEP_SIZE_TRAIN,
    batch_size=BATCH_SIZE,
    validation_data = valid_batches, validation_steps=STEP_SIZE_VALID
)
```

```
27
7
D:\downloads\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.
py:720: UserWarning: This ImageDataGenerator specifies `featurewise_center`, but it has
n't been fit on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
D:\downloads\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.
py:739: UserWarning: This ImageDataGenerator specifies `zca_whitening`, but it hasn't be
en fit on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
Epoch 1/100
27/27 [==============================] - 25s 893ms/step - loss: 0.8419 - acc: 0.6275 - v
al_loss: 0.6343 - val_acc: 0.7292
Epoch 2/100
27/27 [==============================] - 24s 886ms/step - loss: 0.8315 - acc: 0.5901 - v
al_loss: 0.7033 - val_acc: 0.6667
Epoch 3/100
```

```
27/27 [==============================] - 24s 884ms/step - loss: 0.7201 - acc: 0.6891 - v
al_loss: 0.6162 - val_acc: 0.7083
Epoch 4/100
27/27 [==============================] - 24s 904ms/step - loss: 0.7294 - acc: 0.6903 - v
al_loss: 0.7205 - val_acc: 0.6875
Epoch 5/100
27/27 [==============================] - 24s 883ms/step - loss: 0.6578 - acc: 0.6521 - v
al_loss: 0.6881 - val_acc: 0.6458
Epoch 6/100
27/27 [==============================] - 24s 884ms/step - loss: 0.8368 - acc: 0.6195 - v
al_loss: 0.6399 - val_acc: 0.6667
Epoch 7/100
27/27 [==============================] - 24s 885ms/step - loss: 0.7708 - acc: 0.6694 - v
al_loss: 0.7695 - val_acc: 0.6250
Epoch 8/100
27/27 [==============================] - 24s 896ms/step - loss: 0.8226 - acc: 0.6233 - v
al_loss: 0.6963 - val_acc: 0.6875
Epoch 9/100
27/27 [==============================] - 24s 883ms/step - loss: 0.7142 - acc: 0.6678 - v
al_loss: 0.5795 - val_acc: 0.7917
Epoch 10/100
27/27 [==============================] - 24s 891ms/step - loss: 0.7610 - acc: 0.6901 - v
al_loss: 0.6493 - val_acc: 0.7708
Epoch 11/100
27/27 [==============================] - 24s 884ms/step - loss: 0.7458 - acc: 0.6907 - v
al_loss: 0.7509 - val_acc: 0.6875
Epoch 12/100
27/27 [==============================] - 24s 884ms/step - loss: 0.7331 - acc: 0.6856 - v
al_loss: 0.6931 - val_acc: 0.6667
Epoch 13/100
27/27 [==============================] - 24s 889ms/step - loss: 0.7395 - acc: 0.6871 - v
al_loss: 0.6735 - val_acc: 0.6458
Epoch 14/100
27/27 [==============================] - 24s 885ms/step - loss: 0.9057 - acc: 0.5802 - v
al_loss: 0.6172 - val_acc: 0.6875
Epoch 15/100
27/27 [==============================] - 24s 893ms/step - loss: 0.8240 - acc: 0.6699 - v
al_loss: 0.6956 - val_acc: 0.6875
Epoch 16/100
27/27 [==============================] - 24s 882ms/step - loss: 0.8589 - acc: 0.6294 - v
al_loss: 0.6405 - val_acc: 0.6875
Epoch 17/100
27/27 [==============================] - 24s 891ms/step - loss: 0.7685 - acc: 0.6539 - v
al_loss: 0.8279 - val_acc: 0.5208
Epoch 18/100
27/27 [==============================] - 24s 893ms/step - loss: 0.8051 - acc: 0.6592 - v
al_loss: 0.6156 - val_acc: 0.7292
Epoch 19/100
27/27 [==============================] - 24s 893ms/step - loss: 0.8466 - acc: 0.6167 - v
al_loss: 0.8238 - val_acc: 0.5625
Epoch 20/100
27/27 [==============================] - 24s 899ms/step - loss: 0.7716 - acc: 0.6726 - v
al_loss: 0.6417 - val_acc: 0.6667
Epoch 21/100
27/27 [==============================] - 24s 885ms/step - loss: 0.7107 - acc: 0.7169 - v
al_loss: 0.6277 - val_acc: 0.7500
Epoch 22/100
27/27 [==============================] - 24s 888ms/step - loss: 0.8729 - acc: 0.5653 - v
al_loss: 0.6392 - val_acc: 0.7500
Epoch 23/100
27/27 [==============================] - 24s 899ms/step - loss: 0.8168 - acc: 0.7111 - v
al_loss: 0.6442 - val_acc: 0.6458
Epoch 24/100
27/27 [==============================] - 24s 896ms/step - loss: 0.6771 - acc: 0.6994 - v
al_loss: 0.7905 - val_acc: 0.5833
```

```
Epoch 25/100
27/27 [==============================] - 24s 895ms/step - loss: 0.7639 - acc: 0.6528 - v
al_loss: 0.6787 - val_acc: 0.7083
Epoch 26/100
27/27 [==============================] - 24s 897ms/step - loss: 0.6982 - acc: 0.7153 - v
al_loss: 0.6705 - val_acc: 0.6667
Epoch 27/100
27/27 [==============================] - 24s 892ms/step - loss: 0.6888 - acc: 0.7413 - v
al_loss: 0.7203 - val_acc: 0.6875
Epoch 28/100
27/27 [==============================] - 24s 889ms/step - loss: 0.7151 - acc: 0.7017 - v
al_loss: 0.6129 - val_acc: 0.6458
Epoch 29/100
27/27 [==============================] - 24s 890ms/step - loss: 0.5977 - acc: 0.7145 - v
al_loss: 0.7244 - val_acc: 0.6042
Epoch 30/100
27/27 [==============================] - 24s 887ms/step - loss: 0.8973 - acc: 0.5716 - v
al_loss: 0.7661 - val_acc: 0.6042
Epoch 31/100
27/27 [==============================] - 24s 892ms/step - loss: 0.7369 - acc: 0.7121 - v
al_loss: 0.7215 - val_acc: 0.7083
Epoch 32/100
27/27 [==============================] - 24s 896ms/step - loss: 0.8419 - acc: 0.6440 - v
al_loss: 0.6419 - val_acc: 0.7083
Epoch 33/100
27/27 [==============================] - 24s 906ms/step - loss: 0.6595 - acc: 0.7362 - v
al_loss: 0.7299 - val_acc: 0.6875
Epoch 34/100
27/27 [==============================] - 24s 900ms/step - loss: 0.8072 - acc: 0.6058 - v
al_loss: 0.9800 - val_acc: 0.4792
Epoch 35/100
27/27 [==============================] - 24s 889ms/step - loss: 0.7779 - acc: 0.6751 - v
al_loss: 0.7230 - val_acc: 0.6458
Epoch 36/100
27/27 [==============================] - 24s 886ms/step - loss: 0.7211 - acc: 0.6263 - v
al_loss: 0.6367 - val_acc: 0.7292
Epoch 37/100
27/27 [==============================] - 24s 890ms/step - loss: 0.7572 - acc: 0.6422 - v
al_loss: 0.6952 - val_acc: 0.6667
Epoch 38/100
27/27 [==============================] - 24s 894ms/step - loss: 0.7315 - acc: 0.6710 - v
al_loss: 0.6940 - val_acc: 0.6875
Epoch 39/100
27/27 [==============================] - 24s 909ms/step - loss: 0.8004 - acc: 0.6549 - v
al_loss: 0.7731 - val_acc: 0.7083
Epoch 40/100
27/27 [==============================] - 24s 894ms/step - loss: 0.6475 - acc: 0.6930 - v
al_loss: 0.9432 - val_acc: 0.6042
Epoch 41/100
27/27 [==============================] - 24s 900ms/step - loss: 0.6875 - acc: 0.7050 - v
al_loss: 0.6609 - val_acc: 0.7083
Epoch 42/100
27/27 [==============================] - 24s 896ms/step - loss: 0.7533 - acc: 0.6934 - v
al_loss: 0.6819 - val_acc: 0.6250
Epoch 43/100
27/27 [==============================] - 24s 895ms/step - loss: 0.7168 - acc: 0.7153 - v
al_loss: 0.7355 - val_acc: 0.6250
Epoch 44/100
27/27 [==============================] - 24s 894ms/step - loss: 0.7387 - acc: 0.6658 - v
al_loss: 0.7431 - val_acc: 0.6042
Epoch 45/100
27/27 [==============================] - 24s 887ms/step - loss: 0.6254 - acc: 0.7534 - v
al_loss: 0.6896 - val_acc: 0.6458
Epoch 46/100
27/27 [==============================] - 24s 893ms/step - loss: 0.7138 - acc: 0.7611 - v
```

```
al_loss: 0.6983 - val_acc: 0.7083
Epoch 47/100
27/27 [==============================] - 24s 894ms/step - loss: 0.7465 - acc: 0.6793 - v
al_loss: 0.6639 - val_acc: 0.7292
Epoch 48/100
27/27 [==============================] - 24s 892ms/step - loss: 0.7294 - acc: 0.6507 - v
al_loss: 0.6644 - val_acc: 0.6667
Epoch 49/100
27/27 [==============================] - 24s 887ms/step - loss: 0.7769 - acc: 0.6448 - v
al_loss: 0.7437 - val_acc: 0.6875
Epoch 50/100
27/27 [==============================] - 24s 888ms/step - loss: 0.8308 - acc: 0.6518 - v
al_loss: 0.7541 - val_acc: 0.6042
Epoch 51/100
27/27 [==============================] - 24s 898ms/step - loss: 0.7410 - acc: 0.6784 - v
al_loss: 0.5126 - val_acc: 0.7708
Epoch 52/100
27/27 [==============================] - 24s 894ms/step - loss: 0.6741 - acc: 0.7019 - v
al_loss: 0.8472 - val_acc: 0.6042
Epoch 53/100
27/27 [==============================] - 24s 896ms/step - loss: 0.8460 - acc: 0.6263 - v
al_loss: 0.6900 - val_acc: 0.6458
Epoch 54/100
27/27 [==============================] - 24s 892ms/step - loss: 0.7382 - acc: 0.6836 - v
al_loss: 0.8077 - val_acc: 0.6250
Epoch 55/100
27/27 [==============================] - 24s 891ms/step - loss: 0.6268 - acc: 0.7096 - v
al_loss: 0.6914 - val_acc: 0.7083
Epoch 56/100
27/27 [==============================] - 24s 892ms/step - loss: 0.7027 - acc: 0.6951 - v
al_loss: 0.7668 - val_acc: 0.5833
Epoch 57/100
27/27 [==============================] - 24s 891ms/step - loss: 0.6765 - acc: 0.7232 - v
al_loss: 0.6990 - val_acc: 0.6458
Epoch 58/100
27/27 [==============================] - 24s 897ms/step - loss: 0.7227 - acc: 0.7002 - v
al_loss: 0.6889 - val_acc: 0.6667
Epoch 59/100
27/27 [==============================] - 24s 891ms/step - loss: 0.6582 - acc: 0.7179 - v
al_loss: 0.7704 - val_acc: 0.5833
Epoch 60/100
27/27 [==============================] - 24s 882ms/step - loss: 0.8044 - acc: 0.6872 - v
al_loss: 0.7953 - val_acc: 0.5833
Epoch 61/100
27/27 [==============================] - 24s 890ms/step - loss: 0.7116 - acc: 0.7058 - v
al_loss: 0.6783 - val_acc: 0.7708
Epoch 62/100
27/27 [==============================] - 24s 899ms/step - loss: 0.7237 - acc: 0.7301 - v
al_loss: 0.6888 - val_acc: 0.6667
Epoch 63/100
27/27 [==============================] - 24s 891ms/step - loss: 0.6623 - acc: 0.7704 - v
al_loss: 0.6058 - val_acc: 0.7292
Epoch 64/100
27/27 [==============================] - 24s 886ms/step - loss: 0.6242 - acc: 0.6990 - v
al_loss: 0.6818 - val_acc: 0.6667
Epoch 65/100
27/27 [==============================] - 24s 889ms/step - loss: 0.6935 - acc: 0.7172 - v
al_loss: 0.8755 - val_acc: 0.6250
Epoch 66/100
27/27 [==============================] - 24s 885ms/step - loss: 0.7034 - acc: 0.6918 - v
al_loss: 0.7286 - val_acc: 0.6458
Epoch 67/100
27/27 [==============================] - 24s 885ms/step - loss: 0.7740 - acc: 0.6878 - v
al_loss: 0.6160 - val_acc: 0.7500
Epoch 68/100
```

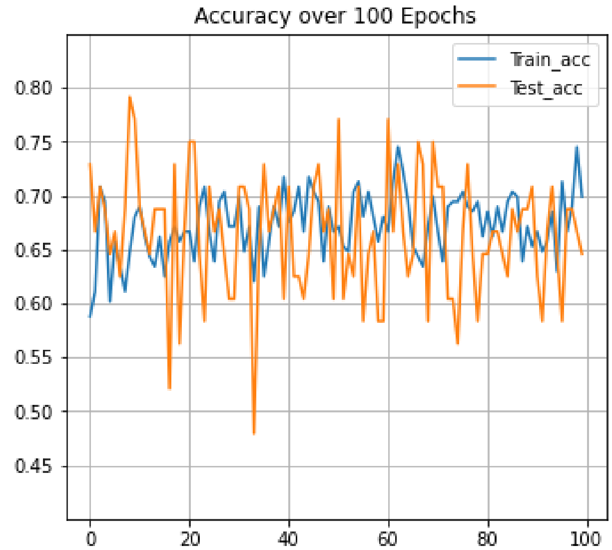
```
27/27 [==============================] - 24s 879ms/step - loss: 0.6955 - acc: 0.6630 - v
al_loss: 0.6424 - val_acc: 0.7292
Epoch 69/100
27/27 [==============================] - 24s 894ms/step - loss: 0.7222 - acc: 0.6820 - v
al_loss: 0.8036 - val_acc: 0.5833
Epoch 70/100
27/27 [==============================] - 24s 879ms/step - loss: 0.7688 - acc: 0.6819 - v
al_loss: 0.6289 - val_acc: 0.7500
Epoch 71/100
27/27 [==============================] - 24s 903ms/step - loss: 0.7886 - acc: 0.6488 - v
al_loss: 0.6324 - val_acc: 0.7083
Epoch 72/100
27/27 [==============================] - 24s 887ms/step - loss: 0.8355 - acc: 0.6151 - v
al_loss: 0.6467 - val_acc: 0.7083
Epoch 73/100
27/27 [==============================] - 24s 887ms/step - loss: 0.7172 - acc: 0.6752 - v
al_loss: 0.7628 - val_acc: 0.6042
Epoch 74/100
27/27 [==============================] - 24s 897ms/step - loss: 0.8222 - acc: 0.6674 - v
al_loss: 0.6299 - val_acc: 0.6042
Epoch 75/100
27/27 [==============================] - 24s 904ms/step - loss: 0.6728 - acc: 0.7331 - v
al_loss: 0.7998 - val_acc: 0.5625
Epoch 76/100
27/27 [==============================] - 24s 887ms/step - loss: 0.8201 - acc: 0.7113 - v
al_loss: 0.7102 - val_acc: 0.6667
Epoch 77/100
27/27 [==============================] - 24s 880ms/step - loss: 0.7638 - acc: 0.6677 - v
al_loss: 0.5557 - val_acc: 0.7292
Epoch 78/100
27/27 [==============================] - 24s 886ms/step - loss: 0.7576 - acc: 0.7023 - v
al_loss: 0.6808 - val_acc: 0.6458
Epoch 79/100
27/27 [==============================] - 24s 883ms/step - loss: 0.8642 - acc: 0.6704 - v
al_loss: 0.7826 - val_acc: 0.5833
Epoch 80/100
27/27 [==============================] - 24s 898ms/step - loss: 0.6294 - acc: 0.6860 - v
al_loss: 0.6781 - val_acc: 0.6458
Epoch 81/100
27/27 [==============================] - 24s 887ms/step - loss: 0.8092 - acc: 0.6765 - v
al_loss: 0.9560 - val_acc: 0.6458
Epoch 82/100
27/27 [==============================] - 24s 894ms/step - loss: 0.7922 - acc: 0.6379 - v
al_loss: 0.6930 - val_acc: 0.6667
Epoch 83/100
27/27 [==============================] - 24s 887ms/step - loss: 0.7283 - acc: 0.6788 - v
al_loss: 0.7118 - val_acc: 0.6667
Epoch 84/100
27/27 [==============================] - 24s 884ms/step - loss: 0.8359 - acc: 0.6436 - v
al_loss: 0.7258 - val_acc: 0.6458
Epoch 85/100
27/27 [==============================] - 24s 890ms/step - loss: 0.6542 - acc: 0.7220 - v
al_loss: 0.6920 - val_acc: 0.6250
Epoch 86/100
27/27 [==============================] - 24s 893ms/step - loss: 0.6985 - acc: 0.6964 - v
al_loss: 0.6510 - val_acc: 0.6875
Epoch 87/100
27/27 [==============================] - 24s 879ms/step - loss: 0.6349 - acc: 0.7227 - v
al_loss: 0.6698 - val_acc: 0.6667
Epoch 88/100
27/27 [==============================] - 24s 881ms/step - loss: 0.7501 - acc: 0.6605 - v
al_loss: 0.7017 - val_acc: 0.6875
Epoch 89/100
27/27 [==============================] - 24s 880ms/step - loss: 0.7049 - acc: 0.6725 - v
al_loss: 0.7317 - val_acc: 0.6875
```

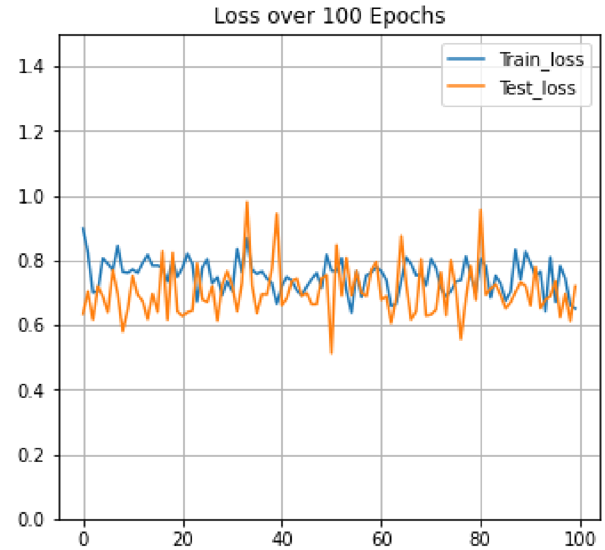
```
Epoch 90/100
27/27 [==============================] - 24s 880ms/step - loss: 0.8217 - acc: 0.6626 - v
al_loss: 0.7210 - val_acc: 0.7083
Epoch 91/100
27/27 [==============================] - 24s 884ms/step - loss: 0.7683 - acc: 0.6941 - v
al_loss: 0.6600 - val_acc: 0.6250
Epoch 92/100
27/27 [==============================] - 24s 892ms/step - loss: 0.7044 - acc: 0.6719 - v
al_loss: 0.7793 - val_acc: 0.5833
Epoch 93/100
27/27 [==============================] - 24s 892ms/step - loss: 0.7508 - acc: 0.6310 - v
al_loss: 0.6521 - val_acc: 0.6667
Epoch 94/100
27/27 [==============================] - 24s 890ms/step - loss: 0.6218 - acc: 0.6899 - v
al_loss: 0.6776 - val_acc: 0.7083
Epoch 95/100
27/27 [==============================] - 24s 885ms/step - loss: 0.8313 - acc: 0.6356 - v
al_loss: 0.6887 - val_acc: 0.6458
Epoch 96/100
27/27 [==============================] - 24s 881ms/step - loss: 0.5924 - acc: 0.7433 - v
al_loss: 0.7365 - val_acc: 0.5833
Epoch 97/100
27/27 [==============================] - 24s 898ms/step - loss: 0.9186 - acc: 0.6458 - v
al_loss: 0.6239 - val_acc: 0.6875
Epoch 98/100
27/27 [==============================] - 24s 898ms/step - loss: 0.6746 - acc: 0.7332 - v
al_loss: 0.6965 - val_acc: 0.6875
Epoch 99/100
27/27 [==============================] - 24s 897ms/step - loss: 0.6423 - acc: 0.7498 - v
al_loss: 0.6115 - val_acc: 0.6667
Epoch 100/100
27/27 [==============================] - 24s 899ms/step - loss: 0.5739 - acc: 0.7325 - v
al_loss: 0.7202 - val_acc: 0.6458
```

## [5 points] Plot Accuracy and Loss During Training

In [30]:

```
import matplotlib.pyplot as plt

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

plt.rcParams['figure.figsize'] = [12, 5]
fig, (ax1, ax2) = plt.subplots(1, 2)


ax1.plot(history.history['acc'], label = 'Train_acc')
ax1.plot(history.history['val_acc'], label = 'Test_acc')
ax1.set_title("Accuracy over 100 Epochs")
ax1.legend(['Train_acc','Test_acc'])
ax1.grid()
ax1.set_yticks([0.45, 0.5,0.55,0.6,0.65,0.7,0.75,0.8])
ax1.set_ylim([0.4,0.85])

ax2.plot(history.history['loss'], label = 'Train_loss')
ax2.plot(history.history['val_loss'], label = 'Test_loss')
ax2.set_title("Loss over 100 Epochs")
ax2.legend(['Train_loss','Test_loss'])
ax2.set_ylim([0,1.5])
ax2.grid()


plt.show()
```




## Testing Model

In [ ]:
```
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,cla
eval_generator.reset()
print(len(eval_generator))
```

In [19]:
```
#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
D:\downloads\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:187
7: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future
version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '
36/36 [==============================] - 4s 94ms/step - loss: 0.8552 - acc: 0.5833
Test loss: 0.8551800847053528
Test accuracy: 0.5833333134651184
```

In [34]:
```
x6 = model6.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
                           use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x6[0])
print('Test accuracy:',x6[1])
```

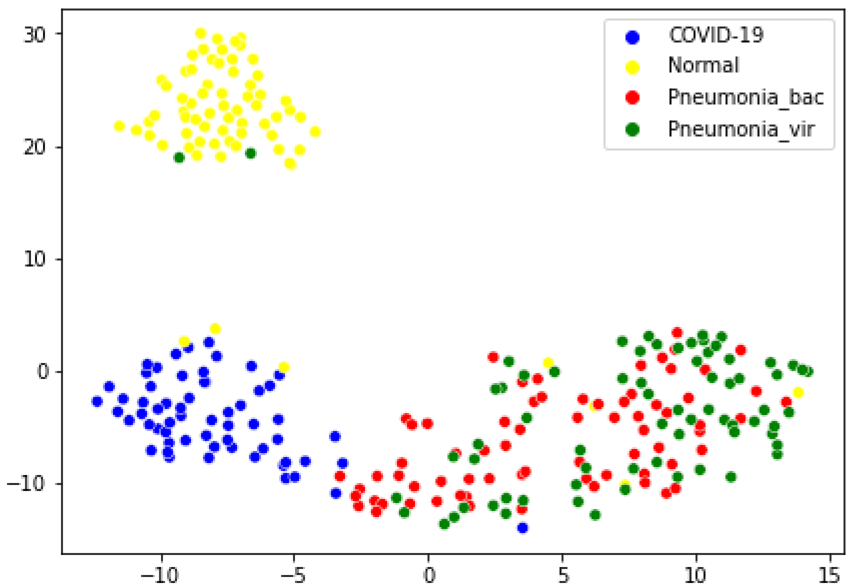
```
36/36 [==============================] - 3s 95ms/step - loss: 0.6917 - acc: 0.7222
Test loss: 0.6917243599891663
Test accuracy: 0.7222222089767456
```

# [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 [36]:
```
from sklearn.manifold import TSNE
import seaborn as sns

intermediate_layer_model = tf.keras.models.Model(inputs=model6.input,
                                        outputs=model6.get_layer('dense_feature').outpu

tsne_eval_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_S
                                                    batch_size=1,shuffle=False,seed=42,cl

#raise NotImplementedError("Extract features from the tsne_data_generator and fit a t-S
#                           "and plot the resulting 2D features of the four classes.")

plt.rcParams['figure.figsize'] = [7, 5]

intermediate_output = intermediate_layer_model.predict(tsne_eval_generator)
tsne = TSNE(n_components=2)
results = tsne.fit_transform(intermediate_output)
h = tsne_eval_generator.labels
sns.scatterplot(x = results[:,0], y = results[:,1], legend = 'full', hue= h, palette=["

L=plt.legend()
L.get_texts()[0].set_text('COVID-19')
L.get_texts()[1].set_text('Normal')
L.get_texts()[2].set_text('Pneumonia_bac')
L.get_texts()[3].set_text('Pneumonia_vir')
```

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



# 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]: '2.4.1'

## Load Image Data

In [7]:
```
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    = 8  # try reducing batch size or freeze more layers if your GPU runs out
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0005 # start off with high rate first 0.001 and experiment with reduc
```

## Generate Training and Validation Batches

In [8]:
```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center
                                   featurewise_std_normalization = True,width_shift_ran
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0
                                   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.
```

## [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 [9]:
```
#raise NotImplementedError("Build your model based on an architecture of your choice "
 #                          "A sample model summary is shown below")

mob_model = tf.keras.applications.MobileNet(
    weights = "imagenet",
    input_shape = (224, 224, 3),
    include_top = False
)

mob_model.trainable = False

model = tf.keras.models.Sequential()
model.add(mob_model)
model.add(tf.keras.layers.AveragePooling2D(pool_size=(3,3)))
```

```
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dropout(rate = 0.5))
model.add(tf.keras.layers.Dense(256, name = "dense_feature"))
model.add(tf.keras.layers.Dense(4, activation='softmax'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
mobilenet_1.00_224 (Function (None, 7, 7, 1024)        3228864   
_________________________________________________________________
average_pooling2d_1 (Average (None, 2, 2, 1024)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0         
_________________________________________________________________
dense_feature (Dense)        (None, 256)               1048832   
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028      
=================================================================
Total params: 4,278,724
Trainable params: 1,049,860
Non-trainable params: 3,228,864
_________________________________________________________________
```

### [5 points] Train Model

In [10]:

```
model.compile(optimizer='adam',
                loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
                metrics=['acc'])

#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

#raise NotImplementedError("Use the model.fit function to train your network")

history = model.fit(
    train_batches,
    epochs=NUM_EPOCHS, steps_per_epoch=STEP_SIZE_TRAIN,
    batch_size=BATCH_SIZE,
    validation_data = valid_batches, validation_steps=STEP_SIZE_VALID
)
```

```
27
7
Epoch 1/100
27/27 [==============================] - 10s 323ms/step - loss: 11.2312 - acc: 0.3311 - val_loss: 3.0569 - val_acc: 0.6667
Epoch 2/100
27/27 [==============================] - 8s 296ms/step - loss: 3.0541 - acc: 0.6738 - val_loss: 3.5744 - val_acc: 0.5625
Epoch 3/100
27/27 [==============================] - 8s 294ms/step - loss: 4.1460 - acc: 0.6065 - val_loss: 4.0156 - val_acc: 0.6458
Epoch 4/100
27/27 [==============================] - 8s 292ms/step - loss: 4.0945 - acc: 0.6594 - val_loss: 5.6033 - val_acc: 0.6042
```

```
Epoch 5/100
27/27 [==============================] - 8s 295ms/step - loss: 4.5603 - acc: 0.6335 - va
l_loss: 4.6616 - val_acc: 0.6458
Epoch 6/100
27/27 [==============================] - 8s 296ms/step - loss: 3.2647 - acc: 0.6697 - va
l_loss: 3.2009 - val_acc: 0.6875
Epoch 7/100
27/27 [==============================] - 8s 297ms/step - loss: 2.3566 - acc: 0.7078 - va
l_loss: 3.0525 - val_acc: 0.6250
Epoch 8/100
27/27 [==============================] - 8s 295ms/step - loss: 2.3730 - acc: 0.7096 - va
l_loss: 2.2620 - val_acc: 0.6042
Epoch 9/100
27/27 [==============================] - 8s 289ms/step - loss: 2.1428 - acc: 0.7233 - va
l_loss: 1.9770 - val_acc: 0.5833
Epoch 10/100
27/27 [==============================] - 8s 292ms/step - loss: 2.3856 - acc: 0.6956 - va
l_loss: 1.6598 - val_acc: 0.6458
Epoch 11/100
27/27 [==============================] - 8s 299ms/step - loss: 2.0523 - acc: 0.7196 - va
l_loss: 1.2245 - val_acc: 0.7083
Epoch 12/100
27/27 [==============================] - 8s 296ms/step - loss: 1.1680 - acc: 0.7906 - va
l_loss: 2.5238 - val_acc: 0.6250
Epoch 13/100
27/27 [==============================] - 8s 292ms/step - loss: 1.2114 - acc: 0.8334 - va
l_loss: 1.7448 - val_acc: 0.6458
Epoch 14/100
27/27 [==============================] - 8s 296ms/step - loss: 1.2012 - acc: 0.7762 - va
l_loss: 1.1534 - val_acc: 0.6458
Epoch 15/100
27/27 [==============================] - 8s 293ms/step - loss: 1.5160 - acc: 0.7145 - va
l_loss: 0.9955 - val_acc: 0.6875
Epoch 16/100
27/27 [==============================] - 8s 297ms/step - loss: 0.8387 - acc: 0.7865 - va
l_loss: 0.6312 - val_acc: 0.7083
Epoch 17/100
27/27 [==============================] - 8s 296ms/step - loss: 0.7594 - acc: 0.7668 - va
l_loss: 0.7860 - val_acc: 0.7083
Epoch 18/100
27/27 [==============================] - 8s 295ms/step - loss: 0.8099 - acc: 0.7549 - va
l_loss: 1.1135 - val_acc: 0.7083
Epoch 19/100
27/27 [==============================] - 8s 296ms/step - loss: 1.0509 - acc: 0.7084 - va
l_loss: 2.5376 - val_acc: 0.5833
Epoch 20/100
27/27 [==============================] - 8s 298ms/step - loss: 0.7945 - acc: 0.8011 - va
l_loss: 1.1394 - val_acc: 0.7083
Epoch 21/100
27/27 [==============================] - 8s 299ms/step - loss: 1.0960 - acc: 0.7227 - va
l_loss: 0.8010 - val_acc: 0.8333
Epoch 22/100
27/27 [==============================] - 8s 291ms/step - loss: 0.9010 - acc: 0.7823 - va
l_loss: 0.8499 - val_acc: 0.6875
Epoch 23/100
27/27 [==============================] - 8s 291ms/step - loss: 0.7447 - acc: 0.7551 - va
l_loss: 0.9075 - val_acc: 0.8125
Epoch 24/100
27/27 [==============================] - 8s 292ms/step - loss: 0.7742 - acc: 0.7721 - va
l_loss: 1.6915 - val_acc: 0.6875
Epoch 25/100
27/27 [==============================] - 8s 296ms/step - loss: 0.7604 - acc: 0.7337 - va
l_loss: 1.3095 - val_acc: 0.6875
Epoch 26/100
27/27 [==============================] - 8s 294ms/step - loss: 0.5772 - acc: 0.8336 - va
```

```
l_loss: 1.2574 - val_acc: 0.6458
Epoch 27/100
27/27 [==============================] - 8s 293ms/step - loss: 0.6558 - acc: 0.7781 - va
l_loss: 1.5179 - val_acc: 0.6667
Epoch 28/100
27/27 [==============================] - 8s 291ms/step - loss: 0.8765 - acc: 0.7308 - va
l_loss: 0.8431 - val_acc: 0.7708
Epoch 29/100
27/27 [==============================] - 8s 293ms/step - loss: 0.6724 - acc: 0.8165 - va
l_loss: 0.8781 - val_acc: 0.7917
Epoch 30/100
27/27 [==============================] - 8s 288ms/step - loss: 0.7340 - acc: 0.7562 - va
l_loss: 0.5890 - val_acc: 0.6667
Epoch 31/100
27/27 [==============================] - 8s 295ms/step - loss: 0.7749 - acc: 0.7641 - va
l_loss: 1.0802 - val_acc: 0.6667
Epoch 32/100
27/27 [==============================] - 8s 296ms/step - loss: 0.6995 - acc: 0.7746 - va
l_loss: 1.7057 - val_acc: 0.6458
Epoch 33/100
27/27 [==============================] - 8s 295ms/step - loss: 0.6776 - acc: 0.7561 - va
l_loss: 0.8243 - val_acc: 0.6667
Epoch 34/100
27/27 [==============================] - 8s 297ms/step - loss: 0.7412 - acc: 0.7768 - va
l_loss: 1.0690 - val_acc: 0.6250
Epoch 35/100
27/27 [==============================] - 8s 308ms/step - loss: 0.5689 - acc: 0.7782 - va
l_loss: 0.7362 - val_acc: 0.7708
Epoch 36/100
27/27 [==============================] - 8s 311ms/step - loss: 0.5269 - acc: 0.8105 - va
l_loss: 0.6282 - val_acc: 0.7708
Epoch 37/100
27/27 [==============================] - 8s 303ms/step - loss: 0.5517 - acc: 0.7402 - va
l_loss: 0.5984 - val_acc: 0.7917
Epoch 38/100
27/27 [==============================] - 8s 295ms/step - loss: 0.4117 - acc: 0.8286 - va
l_loss: 0.6388 - val_acc: 0.7708
Epoch 39/100
27/27 [==============================] - 8s 300ms/step - loss: 0.4453 - acc: 0.7870 - va
l_loss: 0.9935 - val_acc: 0.7083
Epoch 40/100
27/27 [==============================] - 8s 297ms/step - loss: 0.4734 - acc: 0.8332 - va
l_loss: 0.6494 - val_acc: 0.7083
Epoch 41/100
27/27 [==============================] - 8s 300ms/step - loss: 0.4635 - acc: 0.8191 - va
l_loss: 0.9619 - val_acc: 0.6250
Epoch 42/100
27/27 [==============================] - 8s 299ms/step - loss: 0.5217 - acc: 0.8039 - va
l_loss: 0.7039 - val_acc: 0.7083
Epoch 43/100
27/27 [==============================] - 8s 293ms/step - loss: 0.4188 - acc: 0.8433 - va
l_loss: 0.8029 - val_acc: 0.6667
Epoch 44/100
27/27 [==============================] - 8s 295ms/step - loss: 0.5868 - acc: 0.7902 - va
l_loss: 0.7478 - val_acc: 0.6875
Epoch 45/100
27/27 [==============================] - 8s 292ms/step - loss: 0.5350 - acc: 0.7510 - va
l_loss: 0.6239 - val_acc: 0.7083
Epoch 46/100
27/27 [==============================] - 8s 297ms/step - loss: 0.5295 - acc: 0.7857 - va
l_loss: 0.7166 - val_acc: 0.6875
Epoch 47/100
27/27 [==============================] - 8s 302ms/step - loss: 0.5475 - acc: 0.7698 - va
l_loss: 0.8303 - val_acc: 0.6250
Epoch 48/100
```

```
27/27 [==============================] - 8s 295ms/step - loss: 0.5485 - acc: 0.7594 - va
l_loss: 0.4036 - val_acc: 0.8333
Epoch 49/100
27/27 [==============================] - 8s 301ms/step - loss: 0.5463 - acc: 0.7548 - va
l_loss: 0.5986 - val_acc: 0.7292
Epoch 50/100
27/27 [==============================] - 8s 303ms/step - loss: 0.6422 - acc: 0.7652 - va
l_loss: 0.6672 - val_acc: 0.7292
Epoch 51/100
27/27 [==============================] - 9s 325ms/step - loss: 0.4828 - acc: 0.7493 - va
l_loss: 0.6434 - val_acc: 0.6875
Epoch 52/100
27/27 [==============================] - 9s 325ms/step - loss: 0.5168 - acc: 0.7596 - va
l_loss: 1.0131 - val_acc: 0.5833
Epoch 53/100
27/27 [==============================] - 8s 306ms/step - loss: 0.4663 - acc: 0.8553 - va
l_loss: 0.6286 - val_acc: 0.7500
Epoch 54/100
27/27 [==============================] - 8s 303ms/step - loss: 0.3647 - acc: 0.8371 - va
l_loss: 0.4446 - val_acc: 0.7708
Epoch 55/100
27/27 [==============================] - 8s 313ms/step - loss: 0.4588 - acc: 0.8074 - va
l_loss: 0.6856 - val_acc: 0.6667
Epoch 56/100
27/27 [==============================] - 9s 316ms/step - loss: 0.5935 - acc: 0.7530 - va
l_loss: 0.5967 - val_acc: 0.7292
Epoch 57/100
27/27 [==============================] - 8s 304ms/step - loss: 0.5935 - acc: 0.7932 - va
l_loss: 0.7353 - val_acc: 0.6875
Epoch 58/100
27/27 [==============================] - 8s 302ms/step - loss: 0.6515 - acc: 0.7656 - va
l_loss: 0.6026 - val_acc: 0.7083
Epoch 59/100
27/27 [==============================] - 8s 294ms/step - loss: 0.5735 - acc: 0.7760 - va
l_loss: 0.7741 - val_acc: 0.6875
Epoch 60/100
27/27 [==============================] - 8s 297ms/step - loss: 0.5044 - acc: 0.7854 - va
l_loss: 0.5395 - val_acc: 0.8125
Epoch 61/100
27/27 [==============================] - 8s 300ms/step - loss: 0.4948 - acc: 0.7503 - va
l_loss: 0.5659 - val_acc: 0.7500
Epoch 62/100
27/27 [==============================] - 8s 303ms/step - loss: 0.3943 - acc: 0.8193 - va
l_loss: 0.5311 - val_acc: 0.7708
Epoch 63/100
27/27 [==============================] - 8s 312ms/step - loss: 0.5046 - acc: 0.8052 - va
l_loss: 0.8750 - val_acc: 0.6458
Epoch 64/100
27/27 [==============================] - 8s 291ms/step - loss: 0.5119 - acc: 0.7627 - va
l_loss: 0.7868 - val_acc: 0.6875
Epoch 65/100
27/27 [==============================] - 8s 298ms/step - loss: 0.4463 - acc: 0.8244 - va
l_loss: 0.7574 - val_acc: 0.6875
Epoch 66/100
27/27 [==============================] - 8s 299ms/step - loss: 0.4131 - acc: 0.8158 - va
l_loss: 0.8370 - val_acc: 0.6667
Epoch 67/100
27/27 [==============================] - 8s 300ms/step - loss: 0.5376 - acc: 0.7748 - va
l_loss: 0.4540 - val_acc: 0.7500
Epoch 68/100
27/27 [==============================] - 8s 304ms/step - loss: 0.5658 - acc: 0.7857 - va
l_loss: 0.8571 - val_acc: 0.6875
Epoch 69/100
27/27 [==============================] - 8s 307ms/step - loss: 0.4172 - acc: 0.8612 - va
l_loss: 0.5955 - val_acc: 0.7708
```

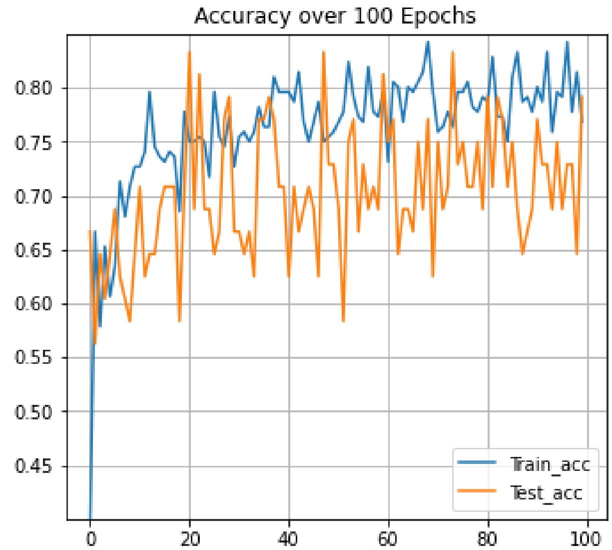
```
Epoch 70/100
27/27 [==============================] - 15s 547ms/step - loss: 0.3621 - acc: 0.8547 - v
al_loss: 0.7286 - val_acc: 0.6250
Epoch 71/100
27/27 [==============================] - 13s 459ms/step - loss: 0.6055 - acc: 0.7285 - v
al_loss: 0.6492 - val_acc: 0.7500
Epoch 72/100
27/27 [==============================] - 18s 653ms/step - loss: 0.5205 - acc: 0.7778 - v
al_loss: 0.6439 - val_acc: 0.6875
Epoch 73/100
27/27 [==============================] - 18s 670ms/step - loss: 0.4689 - acc: 0.7720 - v
al_loss: 0.6271 - val_acc: 0.7083
Epoch 74/100
27/27 [==============================] - 18s 656ms/step - loss: 0.5327 - acc: 0.7744 - v
al_loss: 0.5070 - val_acc: 0.8333
Epoch 75/100
27/27 [==============================] - 18s 657ms/step - loss: 0.5290 - acc: 0.7986 - v
al_loss: 0.6646 - val_acc: 0.7292
Epoch 76/100
27/27 [==============================] - 18s 656ms/step - loss: 0.5619 - acc: 0.7648 - v
al_loss: 0.7399 - val_acc: 0.7500
Epoch 77/100
27/27 [==============================] - 18s 662ms/step - loss: 0.4266 - acc: 0.8110 - v
al_loss: 0.7649 - val_acc: 0.7083
Epoch 78/100
27/27 [==============================] - 18s 656ms/step - loss: 0.5522 - acc: 0.7488 - v
al_loss: 0.5825 - val_acc: 0.7083
Epoch 79/100
27/27 [==============================] - 18s 663ms/step - loss: 0.4236 - acc: 0.7884 - v
al_loss: 0.8876 - val_acc: 0.7500
Epoch 80/100
27/27 [==============================] - 18s 672ms/step - loss: 0.4806 - acc: 0.7989 - v
al_loss: 0.6611 - val_acc: 0.6875
Epoch 81/100
27/27 [==============================] - 17s 631ms/step - loss: 0.5876 - acc: 0.7538 - v
al_loss: 0.5563 - val_acc: 0.7917
Epoch 82/100
27/27 [==============================] - 17s 633ms/step - loss: 0.4501 - acc: 0.8237 - v
al_loss: 0.7729 - val_acc: 0.7083
Epoch 83/100
27/27 [==============================] - 17s 642ms/step - loss: 0.4508 - acc: 0.8068 - v
al_loss: 0.6169 - val_acc: 0.7917
Epoch 84/100
27/27 [==============================] - 17s 645ms/step - loss: 0.4791 - acc: 0.8061 - v
al_loss: 0.4517 - val_acc: 0.7708
Epoch 85/100
27/27 [==============================] - 17s 620ms/step - loss: 0.5523 - acc: 0.7566 - v
al_loss: 0.7037 - val_acc: 0.7083
Epoch 86/100
27/27 [==============================] - 17s 633ms/step - loss: 0.5422 - acc: 0.7866 - v
al_loss: 0.4863 - val_acc: 0.7500
Epoch 87/100
27/27 [==============================] - 17s 628ms/step - loss: 0.6992 - acc: 0.8100 - v
al_loss: 0.5372 - val_acc: 0.6875
Epoch 88/100
27/27 [==============================] - 18s 680ms/step - loss: 0.5421 - acc: 0.7473 - v
al_loss: 0.9470 - val_acc: 0.6458
Epoch 89/100
27/27 [==============================] - 17s 624ms/step - loss: 0.5329 - acc: 0.7797 - v
al_loss: 0.5515 - val_acc: 0.6667
Epoch 90/100
27/27 [==============================] - 17s 635ms/step - loss: 0.4594 - acc: 0.8076 - v
al_loss: 0.5927 - val_acc: 0.6875
Epoch 91/100
27/27 [==============================] - 17s 629ms/step - loss: 0.5164 - acc: 0.8198 - v
```

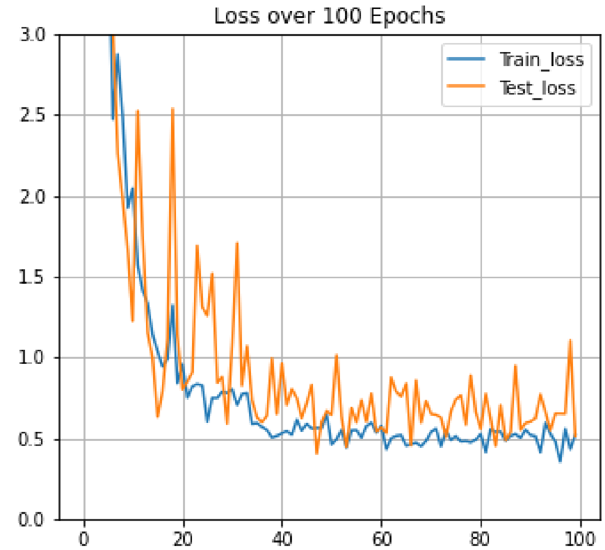
```
al_loss: 0.6016 - val_acc: 0.7708
Epoch 92/100
27/27 [==============================] - 18s 658ms/step - loss: 0.4633 - acc: 0.8018 - v
al_loss: 0.6218 - val_acc: 0.7292
Epoch 93/100
27/27 [==============================] - 17s 641ms/step - loss: 0.3529 - acc: 0.8596 - v
al_loss: 0.7698 - val_acc: 0.7292
Epoch 94/100
27/27 [==============================] - 17s 636ms/step - loss: 0.6671 - acc: 0.7228 - v
al_loss: 0.6634 - val_acc: 0.6875
Epoch 95/100
27/27 [==============================] - 18s 658ms/step - loss: 0.5657 - acc: 0.7655 - v
al_loss: 0.5495 - val_acc: 0.7500
Epoch 96/100
27/27 [==============================] - 17s 645ms/step - loss: 0.5240 - acc: 0.7852 - v
al_loss: 0.6528 - val_acc: 0.6875
Epoch 97/100
27/27 [==============================] - 17s 637ms/step - loss: 0.3751 - acc: 0.8202 - v
al_loss: 0.6523 - val_acc: 0.7292
Epoch 98/100
27/27 [==============================] - 18s 637ms/step - loss: 0.6194 - acc: 0.7432 - v
al_loss: 0.6509 - val_acc: 0.7292
Epoch 99/100
27/27 [==============================] - 17s 612ms/step - loss: 0.4360 - acc: 0.8111 - v
al_loss: 1.1032 - val_acc: 0.6458
Epoch 100/100
27/27 [==============================] - 17s 644ms/step - loss: 0.5366 - acc: 0.7495 - v
al_loss: 0.5136 - val_acc: 0.7917
```

## [5 points] Plot Accuracy and Loss During Training

In [14]:

```python
import matplotlib.pyplot as plt

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

plt.rcParams['figure.figsize'] = [12, 5]
fig, (ax1, ax2) = plt.subplots(1, 2)


ax1.plot(history.history['acc'], label = 'Train_acc')
ax1.plot(history.history['val_acc'], label = 'Test_acc')
ax1.set_title("Accuracy over 100 Epochs")
ax1.legend(['Train_acc','Test_acc'])
ax1.grid()
ax1.set_yticks([0.45, 0.5,0.55,0.6,0.65,0.7,0.75,0.8])
ax1.set_ylim([0.4,0.85])

ax2.plot(history.history['loss'], label = 'Train_loss')
ax2.plot(history.history['val_loss'], label = 'Test_loss')
ax2.set_title("Loss over 100 Epochs")
ax2.legend(['Train_loss','Test_loss'])
ax2.set_ylim([0,3])
ax2.grid()


plt.show()
```




## Testing Model

In [20]:
```
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,cla
eval_generator.reset()
print(len(eval_generator))
x = model.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
                           use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 2s 55ms/step - loss: 0.8847 - acc: 0.5556
Test loss: 0.8847371935844421
Test accuracy: 0.5555555820465088
```

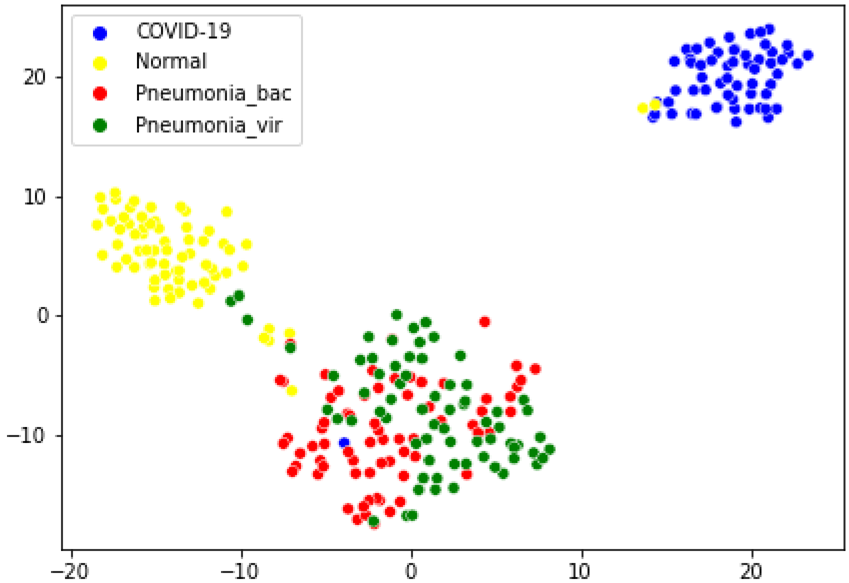
# [10 points] TSNE Plot

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 [18]:
```
from sklearn.manifold import TSNE
import seaborn as sns

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

tsne_eval_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_S
                                                  batch_size=1,shuffle=False,seed=42,cl

#raise NotImplementedError("Extract features from the tsne_data_generator and fit a t-S
#                           "and plot the resulting 2D features of the four classes.")

plt.rcParams['figure.figsize'] = [7, 5]
```

```
intermediate_output = intermediate_layer_model.predict(tsne_eval_generator)
tsne = TSNE(n_components=2)
results = tsne.fit_transform(intermediate_output)
h = tsne_eval_generator.labels
sns.scatterplot(x = results[:,0], y = results[:,1], legend = 'full', hue= h, palette=["

L=plt.legend()
L.get_texts()[0].set_text('COVID-19')
L.get_texts()[1].set_text('Normal')
L.get_texts()[2].set_text('Pneumonia_bac')
L.get_texts()[3].set_text('Pneumonia_vir')
```

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



# 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]: '2.4.1'

### 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    = 8  # try reducing batch size or freeze more layers if your GPU runs out
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0005 # start off with high rate first 0.001 and experiment with reduc
```

### Generate Training and Validation Batches

In [3]:
```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_center
                                   featurewise_std_normalization = True,width_shift_ran
                                   height_shift_range=0.2,shear_range=0.25,zoom_range=0
                                   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.
D:\downloads\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.
py:342: UserWarning: This ImageDataGenerator specifies `zca_whitening` which overrides s
etting 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]:
```
#raise NotImplementedError("Build your model based on an architecture of your choice "
#                          "A sample model summary is shown below")

DenseNet_model = tf.keras.applications.DenseNet121(
    weights = "imagenet",
    input_shape = (224, 224, 3),
    include_top = False
)

DenseNet_model.trainable = False
```

```
model = tf.keras.models.Sequential()
model.add(DenseNet_model)
model.add(tf.keras.layers.AveragePooling2D(pool_size=(3,3)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dropout(rate = 0.5))
model.add(tf.keras.layers.Dense(256, name = "dense_feature"))
model.add(tf.keras.layers.Dense(4, activation='softmax'))
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/dense
net/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5
29089792/29084464 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Functional)     (None, 7, 7, 1024)        7037504
_________________________________________________________________
average_pooling2d (AveragePo (None, 2, 2, 1024)        0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_feature (Dense)        (None, 256)               1048832
_________________________________________________________________
dense (Dense)                (None, 4)                 1028
=================================================================
Total params: 8,087,364
Trainable params: 1,049,860
Non-trainable params: 7,037,504
_________________________________________________________________
```

## [5 points] Train Model

In [5]:

```
model.compile(optimizer='adam',
                loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
                metrics=['acc'])

#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

#raise NotImplementedError("Use the model.fit function to train your network")

history = model.fit(
    train_batches,
    epochs=NUM_EPOCHS, steps_per_epoch=STEP_SIZE_TRAIN,
    batch_size=BATCH_SIZE,
    validation_data = valid_batches, validation_steps=STEP_SIZE_VALID
)
```

```
27
7
D:\downloads\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.
py:720: UserWarning: This ImageDataGenerator specifies `featurewise_center`, but it has
n't been fit on any training data. Fit it first by calling `.fit(numpy_data)`.
```

```
  warnings.warn('This ImageDataGenerator specifies '
D:\downloads\anaconda3\lib\site-packages\keras_preprocessing\image\image_data_generator.
py:739: UserWarning: This ImageDataGenerator specifies `zca_whitening`, but it hasn't be
en fit on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
Epoch 1/100
27/27 [==============================] - 33s 888ms/step - loss: 6.3754 - acc: 0.2846 - v
al_loss: 3.6472 - val_acc: 0.5208
Epoch 2/100
27/27 [==============================] - 18s 682ms/step - loss: 3.9911 - acc: 0.4821 - v
al_loss: 5.1257 - val_acc: 0.3542
Epoch 3/100
27/27 [==============================] - 18s 681ms/step - loss: 4.4674 - acc: 0.5234 - v
al_loss: 3.1368 - val_acc: 0.5417
Epoch 4/100
27/27 [==============================] - 19s 695ms/step - loss: 3.1723 - acc: 0.6085 - v
al_loss: 2.8029 - val_acc: 0.6250
Epoch 5/100
27/27 [==============================] - 19s 685ms/step - loss: 3.1735 - acc: 0.5789 - v
al_loss: 2.5883 - val_acc: 0.6042
Epoch 6/100
27/27 [==============================] - 18s 682ms/step - loss: 2.0017 - acc: 0.5924 - v
al_loss: 3.0670 - val_acc: 0.5833
Epoch 7/100
27/27 [==============================] - 19s 681ms/step - loss: 2.5531 - acc: 0.5824 - v
al_loss: 2.3712 - val_acc: 0.5625
Epoch 8/100
27/27 [==============================] - 18s 673ms/step - loss: 1.9685 - acc: 0.6575 - v
al_loss: 2.2886 - val_acc: 0.4792
Epoch 9/100
27/27 [==============================] - 19s 689ms/step - loss: 1.8148 - acc: 0.7018 - v
al_loss: 1.7604 - val_acc: 0.6042
Epoch 10/100
27/27 [==============================] - 18s 676ms/step - loss: 2.6765 - acc: 0.5635 - v
al_loss: 1.2414 - val_acc: 0.6667
Epoch 11/100
27/27 [==============================] - 18s 666ms/step - loss: 1.7684 - acc: 0.6053 - v
al_loss: 1.0837 - val_acc: 0.6250
Epoch 12/100
27/27 [==============================] - 19s 683ms/step - loss: 1.4519 - acc: 0.6371 - v
al_loss: 2.3088 - val_acc: 0.6042
Epoch 13/100
27/27 [==============================] - 18s 668ms/step - loss: 1.7741 - acc: 0.6756 - v
al_loss: 1.1536 - val_acc: 0.6250
Epoch 14/100
27/27 [==============================] - 18s 668ms/step - loss: 1.5823 - acc: 0.6039 - v
al_loss: 0.8685 - val_acc: 0.7292
Epoch 15/100
27/27 [==============================] - 18s 660ms/step - loss: 0.9587 - acc: 0.7374 - v
al_loss: 1.9798 - val_acc: 0.5833
Epoch 16/100
27/27 [==============================] - 18s 676ms/step - loss: 1.4472 - acc: 0.6512 - v
al_loss: 2.2057 - val_acc: 0.5833
Epoch 17/100
27/27 [==============================] - 19s 680ms/step - loss: 1.1575 - acc: 0.6279 - v
al_loss: 1.2250 - val_acc: 0.6875
Epoch 18/100
27/27 [==============================] - 18s 676ms/step - loss: 1.4207 - acc: 0.5918 - v
al_loss: 0.7621 - val_acc: 0.7083
Epoch 19/100
27/27 [==============================] - 18s 667ms/step - loss: 1.1796 - acc: 0.7049 - v
al_loss: 0.5444 - val_acc: 0.7708
Epoch 20/100
27/27 [==============================] - 18s 679ms/step - loss: 0.9235 - acc: 0.6494 - v
al_loss: 0.8058 - val_acc: 0.6875
```

```
Epoch 21/100
27/27 [==============================] - 18s 679ms/step - loss: 1.0491 - acc: 0.7163 - v
al_loss: 0.8391 - val_acc: 0.7083
Epoch 22/100
27/27 [==============================] - 18s 674ms/step - loss: 0.8957 - acc: 0.7488 - v
al_loss: 0.7248 - val_acc: 0.6875
Epoch 23/100
27/27 [==============================] - 18s 674ms/step - loss: 0.7515 - acc: 0.7286 - v
al_loss: 1.1286 - val_acc: 0.6042
Epoch 24/100
27/27 [==============================] - 18s 674ms/step - loss: 1.0013 - acc: 0.6831 - v
al_loss: 0.7784 - val_acc: 0.7708
Epoch 25/100
27/27 [==============================] - 18s 666ms/step - loss: 1.1342 - acc: 0.6876 - v
al_loss: 0.8484 - val_acc: 0.6667
Epoch 26/100
27/27 [==============================] - 18s 682ms/step - loss: 0.9513 - acc: 0.6819 - v
al_loss: 0.6132 - val_acc: 0.6875
Epoch 27/100
27/27 [==============================] - 18s 679ms/step - loss: 0.6700 - acc: 0.6667 - v
al_loss: 0.7401 - val_acc: 0.5625
Epoch 28/100
27/27 [==============================] - 18s 664ms/step - loss: 0.9512 - acc: 0.6964 - v
al_loss: 0.5261 - val_acc: 0.7292
Epoch 29/100
27/27 [==============================] - 17s 609ms/step - loss: 0.6694 - acc: 0.7209 - v
al_loss: 0.7659 - val_acc: 0.6042
Epoch 30/100
27/27 [==============================] - 15s 542ms/step - loss: 0.7329 - acc: 0.7698 - v
al_loss: 0.6976 - val_acc: 0.6458
Epoch 31/100
27/27 [==============================] - 15s 541ms/step - loss: 0.7671 - acc: 0.6865 - v
al_loss: 1.0127 - val_acc: 0.6250
Epoch 32/100
27/27 [==============================] - 15s 543ms/step - loss: 0.8120 - acc: 0.7093 - v
al_loss: 0.6831 - val_acc: 0.6458
Epoch 33/100
27/27 [==============================] - 15s 540ms/step - loss: 0.7305 - acc: 0.7361 - v
al_loss: 0.6677 - val_acc: 0.6875
Epoch 34/100
27/27 [==============================] - 15s 563ms/step - loss: 0.6613 - acc: 0.6919 - v
al_loss: 0.6181 - val_acc: 0.6667
Epoch 35/100
27/27 [==============================] - 15s 571ms/step - loss: 0.6201 - acc: 0.7346 - v
al_loss: 0.6331 - val_acc: 0.7292
Epoch 36/100
27/27 [==============================] - 16s 581ms/step - loss: 0.9394 - acc: 0.6757 - v
al_loss: 0.6144 - val_acc: 0.7292
Epoch 37/100
27/27 [==============================] - 15s 553ms/step - loss: 0.7278 - acc: 0.6884 - v
al_loss: 0.6354 - val_acc: 0.7917
Epoch 38/100
27/27 [==============================] - 15s 557ms/step - loss: 0.6815 - acc: 0.6858 - v
al_loss: 0.5742 - val_acc: 0.7083
Epoch 39/100
27/27 [==============================] - 15s 551ms/step - loss: 0.7491 - acc: 0.6477 - v
al_loss: 0.7236 - val_acc: 0.6667
Epoch 40/100
27/27 [==============================] - 15s 564ms/step - loss: 0.6387 - acc: 0.7695 - v
al_loss: 0.6195 - val_acc: 0.6875
Epoch 41/100
27/27 [==============================] - 15s 547ms/step - loss: 0.7194 - acc: 0.6955 - v
al_loss: 0.6240 - val_acc: 0.6875
Epoch 42/100
27/27 [==============================] - 15s 546ms/step - loss: 0.6582 - acc: 0.7087 - v
```

```
al_loss: 0.6195 - val_acc: 0.7083
Epoch 43/100
27/27 [==============================] - 15s 545ms/step - loss: 0.6380 - acc: 0.7173 - v
al_loss: 0.7090 - val_acc: 0.6667
Epoch 44/100
27/27 [==============================] - 15s 558ms/step - loss: 0.6449 - acc: 0.7374 - v
al_loss: 0.7256 - val_acc: 0.6250
Epoch 45/100
27/27 [==============================] - 15s 563ms/step - loss: 0.5901 - acc: 0.7954 - v
al_loss: 0.5624 - val_acc: 0.7292
Epoch 46/100
27/27 [==============================] - 15s 559ms/step - loss: 0.8948 - acc: 0.6908 - v
al_loss: 0.7737 - val_acc: 0.6458
Epoch 47/100
27/27 [==============================] - 15s 536ms/step - loss: 0.8152 - acc: 0.6823 - v
al_loss: 0.7177 - val_acc: 0.6667
Epoch 48/100
27/27 [==============================] - 15s 555ms/step - loss: 0.7158 - acc: 0.6870 - v
al_loss: 0.6537 - val_acc: 0.7917
Epoch 49/100
27/27 [==============================] - 15s 559ms/step - loss: 0.6504 - acc: 0.7015 - v
al_loss: 0.5535 - val_acc: 0.6667
Epoch 50/100
27/27 [==============================] - 16s 601ms/step - loss: 0.6654 - acc: 0.7239 - v
al_loss: 0.9537 - val_acc: 0.6042
Epoch 51/100
27/27 [==============================] - 16s 574ms/step - loss: 0.6522 - acc: 0.7079 - v
al_loss: 0.8342 - val_acc: 0.6458
Epoch 52/100
27/27 [==============================] - 15s 559ms/step - loss: 0.6285 - acc: 0.7380 - v
al_loss: 0.7062 - val_acc: 0.6667
Epoch 53/100
27/27 [==============================] - 16s 581ms/step - loss: 0.5916 - acc: 0.7459 - v
al_loss: 0.7048 - val_acc: 0.7500
Epoch 54/100
27/27 [==============================] - 16s 586ms/step - loss: 0.5941 - acc: 0.7486 - v
al_loss: 0.6676 - val_acc: 0.7500
Epoch 55/100
27/27 [==============================] - 16s 590ms/step - loss: 0.7505 - acc: 0.6894 - v
al_loss: 0.5668 - val_acc: 0.7083
Epoch 56/100
27/27 [==============================] - 15s 559ms/step - loss: 0.5605 - acc: 0.7613 - v
al_loss: 0.8058 - val_acc: 0.6458
Epoch 57/100
27/27 [==============================] - 15s 568ms/step - loss: 0.5906 - acc: 0.7538 - v
al_loss: 0.6949 - val_acc: 0.7083
Epoch 58/100
27/27 [==============================] - 15s 569ms/step - loss: 0.6248 - acc: 0.7616 - v
al_loss: 0.5481 - val_acc: 0.7500
Epoch 59/100
27/27 [==============================] - 15s 565ms/step - loss: 0.5763 - acc: 0.7534 - v
al_loss: 0.7867 - val_acc: 0.6250
Epoch 60/100
27/27 [==============================] - 15s 560ms/step - loss: 0.5268 - acc: 0.7664 - v
al_loss: 0.6047 - val_acc: 0.6458
Epoch 61/100
27/27 [==============================] - 19s 714ms/step - loss: 0.6119 - acc: 0.7105 - v
al_loss: 0.5630 - val_acc: 0.7500
Epoch 62/100
27/27 [==============================] - 15s 562ms/step - loss: 0.5412 - acc: 0.7719 - v
al_loss: 0.7738 - val_acc: 0.6875
Epoch 63/100
27/27 [==============================] - 15s 566ms/step - loss: 0.6349 - acc: 0.7143 - v
al_loss: 0.4723 - val_acc: 0.7708
Epoch 64/100
```

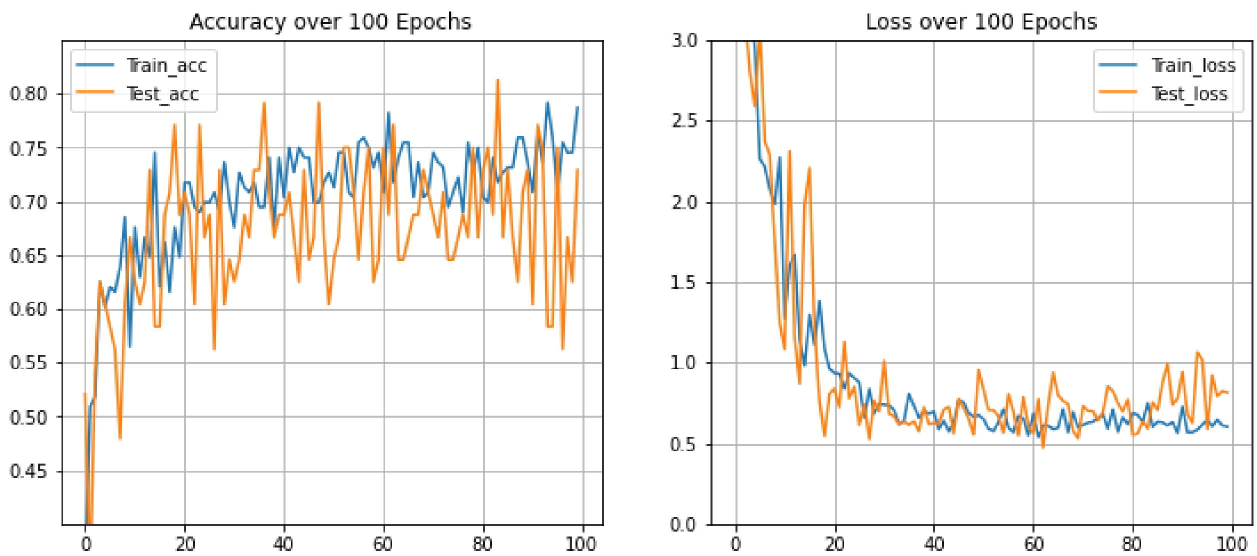
```
27/27 [==============================] - 15s 563ms/step - loss: 0.6017 - acc: 0.7676 - v
al_loss: 0.7356 - val_acc: 0.6458
Epoch 65/100
27/27 [==============================] - 15s 543ms/step - loss: 0.6115 - acc: 0.7658 - v
al_loss: 0.9391 - val_acc: 0.6458
Epoch 66/100
27/27 [==============================] - 15s 556ms/step - loss: 0.5109 - acc: 0.7857 - v
al_loss: 0.7958 - val_acc: 0.6667
Epoch 67/100
27/27 [==============================] - 15s 549ms/step - loss: 0.6900 - acc: 0.6952 - v
al_loss: 0.7647 - val_acc: 0.6875
Epoch 68/100
27/27 [==============================] - 15s 545ms/step - loss: 0.6041 - acc: 0.7138 - v
al_loss: 0.7422 - val_acc: 0.6875
Epoch 69/100
27/27 [==============================] - 15s 554ms/step - loss: 0.7708 - acc: 0.6735 - v
al_loss: 0.5768 - val_acc: 0.7292
Epoch 70/100
27/27 [==============================] - 15s 554ms/step - loss: 0.6501 - acc: 0.6629 - v
al_loss: 0.5303 - val_acc: 0.7083
Epoch 71/100
27/27 [==============================] - 18s 651ms/step - loss: 0.5674 - acc: 0.7653 - v
al_loss: 0.7313 - val_acc: 0.6875
Epoch 72/100
27/27 [==============================] - 17s 632ms/step - loss: 0.7129 - acc: 0.7411 - v
al_loss: 0.7008 - val_acc: 0.6667
Epoch 73/100
27/27 [==============================] - 16s 597ms/step - loss: 0.5912 - acc: 0.7569 - v
al_loss: 0.6984 - val_acc: 0.7083
Epoch 74/100
27/27 [==============================] - 15s 567ms/step - loss: 0.6609 - acc: 0.7131 - v
al_loss: 0.6403 - val_acc: 0.6458
Epoch 75/100
27/27 [==============================] - 16s 575ms/step - loss: 0.5568 - acc: 0.7668 - v
al_loss: 0.6898 - val_acc: 0.6458
Epoch 76/100
27/27 [==============================] - 16s 584ms/step - loss: 0.5815 - acc: 0.7392 - v
al_loss: 0.8531 - val_acc: 0.6667
Epoch 77/100
27/27 [==============================] - 16s 592ms/step - loss: 0.6675 - acc: 0.7117 - v
al_loss: 0.8232 - val_acc: 0.6875
Epoch 78/100
27/27 [==============================] - 16s 609ms/step - loss: 0.6133 - acc: 0.7070 - v
al_loss: 0.7496 - val_acc: 0.6667
Epoch 79/100
27/27 [==============================] - 16s 582ms/step - loss: 0.6972 - acc: 0.7082 - v
al_loss: 0.6965 - val_acc: 0.7500
Epoch 80/100
27/27 [==============================] - 16s 569ms/step - loss: 0.5813 - acc: 0.7615 - v
al_loss: 0.7696 - val_acc: 0.6667
Epoch 81/100
27/27 [==============================] - 15s 567ms/step - loss: 0.7723 - acc: 0.6829 - v
al_loss: 0.5507 - val_acc: 0.7292
Epoch 82/100
27/27 [==============================] - 16s 577ms/step - loss: 0.8333 - acc: 0.7017 - v
al_loss: 0.5607 - val_acc: 0.7500
Epoch 83/100
27/27 [==============================] - 15s 562ms/step - loss: 0.5629 - acc: 0.7899 - v
al_loss: 0.6334 - val_acc: 0.6875
Epoch 84/100
27/27 [==============================] - 15s 568ms/step - loss: 0.6480 - acc: 0.7318 - v
al_loss: 0.5921 - val_acc: 0.8125
Epoch 85/100
27/27 [==============================] - 16s 576ms/step - loss: 0.6523 - acc: 0.6654 - v
al_loss: 0.7556 - val_acc: 0.6667
```

```
Epoch 86/100
27/27 [==============================] - 15s 562ms/step - loss: 0.6343 - acc: 0.7316 - v
al_loss: 0.7051 - val_acc: 0.7292
Epoch 87/100
27/27 [==============================] - 15s 567ms/step - loss: 0.7621 - acc: 0.6934 - v
al_loss: 0.8784 - val_acc: 0.6667
Epoch 88/100
27/27 [==============================] - 15s 547ms/step - loss: 0.6006 - acc: 0.7627 - v
al_loss: 0.9924 - val_acc: 0.6250
Epoch 89/100
27/27 [==============================] - 15s 538ms/step - loss: 0.6824 - acc: 0.7164 - v
al_loss: 0.7398 - val_acc: 0.7083
Epoch 90/100
27/27 [==============================] - 16s 576ms/step - loss: 0.6169 - acc: 0.7004 - v
al_loss: 0.7773 - val_acc: 0.7292
Epoch 91/100
27/27 [==============================] - 16s 580ms/step - loss: 0.8088 - acc: 0.6773 - v
al_loss: 0.9423 - val_acc: 0.6042
Epoch 92/100
27/27 [==============================] - 15s 544ms/step - loss: 0.5230 - acc: 0.7859 - v
al_loss: 0.6786 - val_acc: 0.7708
Epoch 93/100
27/27 [==============================] - 16s 574ms/step - loss: 0.5914 - acc: 0.7190 - v
al_loss: 0.6264 - val_acc: 0.7500
Epoch 94/100
27/27 [==============================] - 16s 585ms/step - loss: 0.6151 - acc: 0.7656 - v
al_loss: 1.0658 - val_acc: 0.5833
Epoch 95/100
27/27 [==============================] - 16s 579ms/step - loss: 0.6008 - acc: 0.7715 - v
al_loss: 1.0154 - val_acc: 0.5833
Epoch 96/100
27/27 [==============================] - 16s 603ms/step - loss: 0.6489 - acc: 0.7106 - v
al_loss: 0.5900 - val_acc: 0.7500
Epoch 97/100
27/27 [==============================] - 15s 570ms/step - loss: 0.5653 - acc: 0.7419 - v
al_loss: 0.9214 - val_acc: 0.5625
Epoch 98/100
27/27 [==============================] - 15s 560ms/step - loss: 0.5790 - acc: 0.7752 - v
al_loss: 0.7909 - val_acc: 0.6667
Epoch 99/100
27/27 [==============================] - 15s 567ms/step - loss: 0.5758 - acc: 0.7348 - v
al_loss: 0.8234 - val_acc: 0.6250
Epoch 100/100
27/27 [==============================] - 15s 568ms/step - loss: 0.6260 - acc: 0.7854 - v
al_loss: 0.8146 - val_acc: 0.7292
```

## [5 points] Plot Accuracy and Loss During Training

In [6]:
```
import matplotlib.pyplot as plt

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

plt.rcParams['figure.figsize'] = [12, 5]
fig, (ax1, ax2) = plt.subplots(1, 2)


ax1.plot(history.history['acc'], label = 'Train_acc')
ax1.plot(history.history['val_acc'], label = 'Test_acc')
ax1.set_title("Accuracy over 100 Epochs")
ax1.legend(['Train_acc','Test_acc'])
ax1.grid()
ax1.set_yticks([0.45, 0.5,0.55,0.6,0.65,0.7,0.75,0.8])
ax1.set_ylim([0.4,0.85])
```

```
ax2.plot(history.history['loss'], label = 'Train_loss')
ax2.plot(history.history['val_loss'], label = 'Test_loss')
ax2.set_title("Loss over 100 Epochs")
ax2.legend(['Train_loss','Test_loss'])
ax2.set_ylim([0,3])
ax2.grid()


plt.show()
```




## Testing Model

In [8]:
```
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,cla
eval_generator.reset()
print(len(eval_generator))
x = model.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator)),
                           use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 3s 74ms/step - loss: 0.7143 - acc: 0.6667
Test loss: 0.7143376469612122
Test accuracy: 0.6666666865348816
```

# [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 [9]:
```
from sklearn.manifold import TSNE
import seaborn as sns

intermediate_layer_model = tf.keras.models.Model(inputs=model.input,
```

```
                                    outputs=model.get_layer('dense_feature').output

tsne_eval_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_S
                                          batch_size=1,shuffle=False,seed=42,cl

#raise NotImplementedError("Extract features from the tsne_data_generator and fit a t-S
#                          "and plot the resulting 2D features of the four classes.")

plt.rcParams['figure.figsize'] = [7, 5]

intermediate_output = intermediate_layer_model.predict(tsne_eval_generator)
tsne = TSNE(n_components=2)
results = tsne.fit_transform(intermediate_output)
h = tsne_eval_generator.labels
sns.scatterplot(x = results[:,0], y = results[:,1], legend = 'full', hue= h, palette=["

L=plt.legend()
L.get_texts()[0].set_text('COVID-19')
L.get_texts()[1].set_text('Normal')
L.get_texts()[2].set_text('Pneumonia_bac')
L.get_texts()[3].set_text('Pneumonia_vir')
```

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