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 (https://drive.google.com/file/d/1Y88tgqpQ1Pjko_7rntcPowOJs_QNOrJ-/view).

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

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

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

[20 points] 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_DEVICE_ORDER"]="PCI_BUS_ID"   # see issue #152
# os.environ["CUDA_VISIBLE_DEVICES"]="0"
tf.__version__
```

Out[1]:

```
'2.6.0'
```

In [2]:

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

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
Num GPUs Available:  1
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 10053575647669565555
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 6252920832
locality {
  bus_id: 1
  links {
  }
}
incarnation: 6610614899885848118
physical_device_desc: "device: 0, name: NVIDIA GeForce RTX 2080 SUPE
R, pci bus id: 0000:01:00.0, compute capability: 7.5"
]
```

Load Image Data

In [3]:

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

Generate Training and Validation Batches

In [4]:

```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_
center = True,
                                   featurewise_std_normalization = True,width_sh
ift_range=0.2,
                                   height_shift_range=0.2,shear_range=0.25,zoom_
range=0.1,
                                   zca_whitening = True,channel_shift_range = 20
,
                                   horizontal_flip = True,vertical_flip = True,
                                   validation_split = 0.2,fill_mode='constant')


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

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

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

```
C:\Users\Li\anaconda3\lib\site-packages\keras_preprocessing\image\im
age_data_generator.py:342: UserWarning: This ImageDataGenerator spec
ifies `zca_whitening` which overrides setting of`featurewise_std_nor
malization`.
  warnings.warn('This ImageDataGenerator specifies '
```

[10 points] Build Model

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

In [5]:

```
from tensorflow.keras.applications import VGG16
from keras.layers.core import Flatten, Dense, Dropout, Lambda


vgg16 = VGG16(weights='imagenet',include_top=False,pooling = "None", classes = 2
, input_shape=(224, 224, 3))
vgg16.trainable = False

model = tf.keras.models.Sequential()
model.add(vgg16)
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dense(4, activation='softmax'))
model.summary()

model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Ada
m(learning_rate=LEARNING_RATE), metrics=['accuracy'])
```

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

[5 points] Train Model

In [6]:

```
#FIT MODEL
print(len(train_batches))
print(len(valid_batches))

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

history=model.fit_generator(train_batches, steps_per_epoch =STEP_SIZE_TRAIN, val
idation_data = valid_batches, validation_steps = STEP_SIZE_VALID, epochs= NUM_EP
OCHS)
```

```
22
6

C:\Users\Li\anaconda3\lib\site-packages\keras\engine\training.py:197
2: UserWarning: `Model.fit_generator` is deprecated and will be remo
ved in a future version. Please use `Model.fit`, which supports gene
rators.
  warnings.warn('`Model.fit_generator` is deprecated and '
C:\Users\Li\anaconda3\lib\site-packages\keras_preprocessing\image\im
age_data_generator.py:720: UserWarning: This ImageDataGenerator spec
ifies `featurewise_center`, but it hasn't been fit on any training d
ata. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
C:\Users\Li\anaconda3\lib\site-packages\keras_preprocessing\image\im
age_data_generator.py:739: UserWarning: This ImageDataGenerator spec
ifies `zca_whitening`, but it hasn't been fit on any training data.
Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '

Epoch 1/100
21/21 [==============================] - 8s 269ms/step - loss: 1.426
4 - accuracy: 0.3010 - val_loss: 1.1429 - val_accuracy: 0.4000
Epoch 2/100
21/21 [==============================] - 5s 237ms/step - loss: 1.173
2 - accuracy: 0.4563 - val_loss: 1.1100 - val_accuracy: 0.4400
Epoch 3/100
21/21 [==============================] - 5s 230ms/step - loss: 1.118
0 - accuracy: 0.4563 - val_loss: 1.1234 - val_accuracy: 0.4600
Epoch 4/100
21/21 [==============================] - 5s 228ms/step - loss: 0.952
3 - accuracy: 0.6019 - val_loss: 0.8814 - val_accuracy: 0.6000
Epoch 5/100
21/21 [==============================] - 5s 234ms/step - loss: 0.960
5 - accuracy: 0.5680 - val_loss: 0.8906 - val_accuracy: 0.6400
Epoch 6/100
21/21 [==============================] - 5s 236ms/step - loss: 0.883
1 - accuracy: 0.6214 - val_loss: 0.9134 - val_accuracy: 0.6000
Epoch 7/100
21/21 [==============================] - 5s 234ms/step - loss: 0.880
1 - accuracy: 0.6117 - val_loss: 0.8144 - val_accuracy: 0.6600
Epoch 8/100
21/21 [==============================] - 5s 234ms/step - loss: 0.839
8 - accuracy: 0.6845 - val_loss: 0.8069 - val_accuracy: 0.6400
Epoch 9/100
21/21 [==============================] - 5s 237ms/step - loss: 0.855
4 - accuracy: 0.6602 - val_loss: 0.7314 - val_accuracy: 0.7200
Epoch 10/100
21/21 [==============================] - 5s 231ms/step - loss: 0.801
2 - accuracy: 0.6359 - val_loss: 0.7197 - val_accuracy: 0.7400
Epoch 11/100
21/21 [==============================] - 5s 230ms/step - loss: 0.822
5 - accuracy: 0.6456 - val_loss: 0.8817 - val_accuracy: 0.5600
Epoch 12/100
21/21 [==============================] - 5s 231ms/step - loss: 0.839
```

```
1 - accuracy: 0.6214 - val_loss: 0.8488 - val_accuracy: 0.5800
Epoch 13/100
21/21 [==============================] - 5s 234ms/step - loss: 0.801
0 - accuracy: 0.6602 - val_loss: 0.7427 - val_accuracy: 0.6200
Epoch 14/100
21/21 [==============================] - 5s 232ms/step - loss: 0.710
3 - accuracy: 0.7039 - val_loss: 0.6787 - val_accuracy: 0.7400
Epoch 15/100
21/21 [==============================] - 5s 235ms/step - loss: 0.675
4 - accuracy: 0.7087 - val_loss: 0.9397 - val_accuracy: 0.5800
Epoch 16/100
21/21 [==============================] - 5s 234ms/step - loss: 0.793
3 - accuracy: 0.6699 - val_loss: 0.7090 - val_accuracy: 0.7000
Epoch 17/100
21/21 [==============================] - 5s 236ms/step - loss: 0.651
5 - accuracy: 0.7087 - val_loss: 0.7031 - val_accuracy: 0.6800
Epoch 18/100
21/21 [==============================] - 5s 234ms/step - loss: 0.687
1 - accuracy: 0.7087 - val_loss: 0.6270 - val_accuracy: 0.7000
Epoch 19/100
21/21 [==============================] - 5s 235ms/step - loss: 0.696
6 - accuracy: 0.7379 - val_loss: 0.8661 - val_accuracy: 0.6600
Epoch 20/100
21/21 [==============================] - 5s 233ms/step - loss: 0.701
6 - accuracy: 0.6699 - val_loss: 0.8272 - val_accuracy: 0.6600
Epoch 21/100
21/21 [==============================] - 5s 234ms/step - loss: 0.727
0 - accuracy: 0.6650 - val_loss: 0.7961 - val_accuracy: 0.5400
Epoch 22/100
21/21 [==============================] - 5s 232ms/step - loss: 0.663
9 - accuracy: 0.7379 - val_loss: 0.8159 - val_accuracy: 0.5800
Epoch 23/100
21/21 [==============================] - 5s 223ms/step - loss: 0.701
2 - accuracy: 0.6990 - val_loss: 0.8038 - val_accuracy: 0.6200
Epoch 24/100
21/21 [==============================] - 5s 231ms/step - loss: 0.727
7 - accuracy: 0.6845 - val_loss: 0.8103 - val_accuracy: 0.6600
Epoch 25/100
21/21 [==============================] - 5s 234ms/step - loss: 0.632
9 - accuracy: 0.7330 - val_loss: 0.7182 - val_accuracy: 0.6600
Epoch 26/100
21/21 [==============================] - 5s 231ms/step - loss: 0.668
0 - accuracy: 0.7184 - val_loss: 0.5410 - val_accuracy: 0.7800
Epoch 27/100
21/21 [==============================] - 5s 234ms/step - loss: 0.613
5 - accuracy: 0.7670 - val_loss: 0.9916 - val_accuracy: 0.5200
Epoch 28/100
21/21 [==============================] - 5s 231ms/step - loss: 0.701
1 - accuracy: 0.6893 - val_loss: 0.9131 - val_accuracy: 0.5200
Epoch 29/100
21/21 [==============================] - 5s 235ms/step - loss: 0.610
7 - accuracy: 0.7476 - val_loss: 0.6755 - val_accuracy: 0.6400
Epoch 30/100
```

```
21/21 [==============================] - 5s 235ms/step - loss: 0.659
1 - accuracy: 0.7136 - val_loss: 0.9429 - val_accuracy: 0.5400
Epoch 31/100
21/21 [==============================] - 5s 235ms/step - loss: 0.679
2 - accuracy: 0.7039 - val_loss: 0.7686 - val_accuracy: 0.7000
Epoch 32/100
21/21 [==============================] - 5s 233ms/step - loss: 0.670
4 - accuracy: 0.7184 - val_loss: 0.7658 - val_accuracy: 0.6800
Epoch 33/100
21/21 [==============================] - 5s 227ms/step - loss: 0.609
5 - accuracy: 0.7573 - val_loss: 0.7125 - val_accuracy: 0.7000
Epoch 34/100
21/21 [==============================] - 5s 226ms/step - loss: 0.645
7 - accuracy: 0.7233 - val_loss: 0.6751 - val_accuracy: 0.7000
Epoch 35/100
21/21 [==============================] - 5s 236ms/step - loss: 0.626
7 - accuracy: 0.7524 - val_loss: 0.7554 - val_accuracy: 0.6400
Epoch 36/100
21/21 [==============================] - 5s 233ms/step - loss: 0.663
2 - accuracy: 0.7282 - val_loss: 0.6110 - val_accuracy: 0.7600
Epoch 37/100
21/21 [==============================] - 5s 230ms/step - loss: 0.597
7 - accuracy: 0.7718 - val_loss: 0.6660 - val_accuracy: 0.6800
Epoch 38/100
21/21 [==============================] - 5s 233ms/step - loss: 0.531
6 - accuracy: 0.7670 - val_loss: 0.7484 - val_accuracy: 0.6000
Epoch 39/100
21/21 [==============================] - 5s 234ms/step - loss: 0.645
1 - accuracy: 0.7282 - val_loss: 0.8453 - val_accuracy: 0.5600
Epoch 40/100
21/21 [==============================] - 5s 235ms/step - loss: 0.642
5 - accuracy: 0.7476 - val_loss: 0.7557 - val_accuracy: 0.6600
Epoch 41/100
21/21 [==============================] - 5s 239ms/step - loss: 0.603
8 - accuracy: 0.7143 - val_loss: 0.6908 - val_accuracy: 0.6400
Epoch 42/100
21/21 [==============================] - 5s 234ms/step - loss: 0.554
3 - accuracy: 0.7670 - val_loss: 0.6859 - val_accuracy: 0.6200
Epoch 43/100
21/21 [==============================] - 5s 234ms/step - loss: 0.592
9 - accuracy: 0.7621 - val_loss: 0.6272 - val_accuracy: 0.7200
Epoch 44/100
21/21 [==============================] - 5s 234ms/step - loss: 0.624
8 - accuracy: 0.7184 - val_loss: 0.6195 - val_accuracy: 0.7200
Epoch 45/100
21/21 [==============================] - 5s 228ms/step - loss: 0.585
0 - accuracy: 0.7670 - val_loss: 0.8039 - val_accuracy: 0.5600
Epoch 46/100
21/21 [==============================] - 5s 230ms/step - loss: 0.537
1 - accuracy: 0.7621 - val_loss: 0.7563 - val_accuracy: 0.5800
Epoch 47/100
21/21 [==============================] - 5s 230ms/step - loss: 0.533
6 - accuracy: 0.7524 - val_loss: 0.7040 - val_accuracy: 0.7000
```

```
Epoch 48/100
21/21 [==============================] - 5s 230ms/step - loss: 0.593
7 - accuracy: 0.7476 - val_loss: 0.6356 - val_accuracy: 0.6800
Epoch 49/100
21/21 [==============================] - 5s 234ms/step - loss: 0.613
2 - accuracy: 0.7233 - val_loss: 0.5993 - val_accuracy: 0.6600
Epoch 50/100
21/21 [==============================] - 5s 229ms/step - loss: 0.633
4 - accuracy: 0.7524 - val_loss: 0.7154 - val_accuracy: 0.7200
Epoch 51/100
21/21 [==============================] - 5s 231ms/step - loss: 0.569
1 - accuracy: 0.7573 - val_loss: 0.6660 - val_accuracy: 0.6800
Epoch 52/100
21/21 [==============================] - 5s 235ms/step - loss: 0.507
8 - accuracy: 0.7961 - val_loss: 0.7096 - val_accuracy: 0.6200
Epoch 53/100
21/21 [==============================] - 5s 231ms/step - loss: 0.532
3 - accuracy: 0.7670 - val_loss: 0.5824 - val_accuracy: 0.7600
Epoch 54/100
21/21 [==============================] - 5s 234ms/step - loss: 0.619
4 - accuracy: 0.7233 - val_loss: 0.6352 - val_accuracy: 0.6600
Epoch 55/100
21/21 [==============================] - 5s 232ms/step - loss: 0.568
4 - accuracy: 0.7816 - val_loss: 0.6507 - val_accuracy: 0.7400
Epoch 56/100
21/21 [==============================] - 5s 235ms/step - loss: 0.542
9 - accuracy: 0.7573 - val_loss: 0.6149 - val_accuracy: 0.6800
Epoch 57/100
21/21 [==============================] - 5s 234ms/step - loss: 0.629
8 - accuracy: 0.7087 - val_loss: 0.8278 - val_accuracy: 0.6200
Epoch 58/100
21/21 [==============================] - 5s 236ms/step - loss: 0.550
4 - accuracy: 0.7573 - val_loss: 0.7283 - val_accuracy: 0.7000
Epoch 59/100
21/21 [==============================] - 5s 232ms/step - loss: 0.539
6 - accuracy: 0.7767 - val_loss: 0.6445 - val_accuracy: 0.7000
Epoch 60/100
21/21 [==============================] - 5s 234ms/step - loss: 0.538
5 - accuracy: 0.7667 - val_loss: 0.7447 - val_accuracy: 0.5800
Epoch 61/100
21/21 [==============================] - 5s 230ms/step - loss: 0.560
0 - accuracy: 0.7816 - val_loss: 0.5994 - val_accuracy: 0.7200
Epoch 62/100
21/21 [==============================] - 5s 236ms/step - loss: 0.490
1 - accuracy: 0.8204 - val_loss: 0.7886 - val_accuracy: 0.5800
Epoch 63/100
21/21 [==============================] - 5s 233ms/step - loss: 0.567
9 - accuracy: 0.7767 - val_loss: 0.7193 - val_accuracy: 0.6200
Epoch 64/100
21/21 [==============================] - 5s 237ms/step - loss: 0.566
6 - accuracy: 0.7427 - val_loss: 0.5512 - val_accuracy: 0.7600
Epoch 65/100
21/21 [==============================] - 5s 236ms/step - loss: 0.554
```

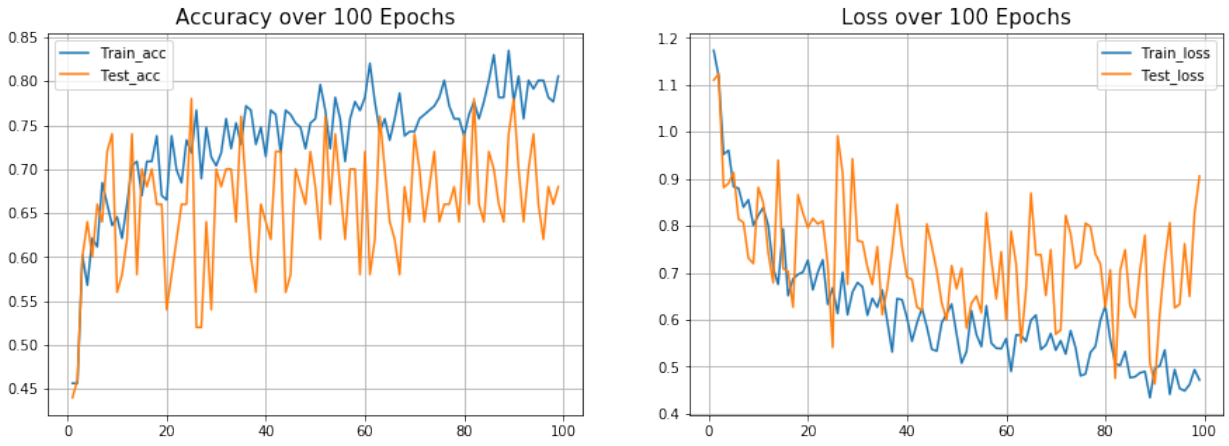
```
4 - accuracy: 0.7573 - val_loss: 0.6635 - val_accuracy: 0.7000
Epoch 66/100
21/21 [==============================] - 5s 230ms/step - loss: 0.599
2 - accuracy: 0.7330 - val_loss: 0.8694 - val_accuracy: 0.6400
Epoch 67/100
21/21 [==============================] - 5s 232ms/step - loss: 0.610
0 - accuracy: 0.7573 - val_loss: 0.7381 - val_accuracy: 0.6200
Epoch 68/100
21/21 [==============================] - 5s 233ms/step - loss: 0.536
7 - accuracy: 0.7864 - val_loss: 0.7385 - val_accuracy: 0.5800
Epoch 69/100
21/21 [==============================] - 5s 232ms/step - loss: 0.546
0 - accuracy: 0.7379 - val_loss: 0.6521 - val_accuracy: 0.6800
Epoch 70/100
21/21 [==============================] - 5s 232ms/step - loss: 0.570
8 - accuracy: 0.7427 - val_loss: 0.7493 - val_accuracy: 0.6400
Epoch 71/100
21/21 [==============================] - 5s 234ms/step - loss: 0.535
3 - accuracy: 0.7427 - val_loss: 0.5695 - val_accuracy: 0.7400
Epoch 72/100
21/21 [==============================] - 5s 234ms/step - loss: 0.555
5 - accuracy: 0.7573 - val_loss: 0.5782 - val_accuracy: 0.7000
Epoch 73/100
21/21 [==============================] - 5s 232ms/step - loss: 0.527
1 - accuracy: 0.7621 - val_loss: 0.8220 - val_accuracy: 0.6400
Epoch 74/100
21/21 [==============================] - 5s 237ms/step - loss: 0.576
9 - accuracy: 0.7670 - val_loss: 0.7831 - val_accuracy: 0.6800
Epoch 75/100
21/21 [==============================] - 5s 235ms/step - loss: 0.540
6 - accuracy: 0.7718 - val_loss: 0.7095 - val_accuracy: 0.7200
Epoch 76/100
21/21 [==============================] - 5s 234ms/step - loss: 0.480
9 - accuracy: 0.7816 - val_loss: 0.7204 - val_accuracy: 0.6400
Epoch 77/100
21/21 [==============================] - 5s 235ms/step - loss: 0.484
6 - accuracy: 0.8010 - val_loss: 0.8055 - val_accuracy: 0.6600
Epoch 78/100
21/21 [==============================] - 5s 232ms/step - loss: 0.530
5 - accuracy: 0.7718 - val_loss: 0.7984 - val_accuracy: 0.6600
Epoch 79/100
21/21 [==============================] - 5s 236ms/step - loss: 0.543
5 - accuracy: 0.7573 - val_loss: 0.7398 - val_accuracy: 0.6800
Epoch 80/100
21/21 [==============================] - 5s 237ms/step - loss: 0.598
5 - accuracy: 0.7573 - val_loss: 0.7189 - val_accuracy: 0.6400
Epoch 81/100
21/21 [==============================] - 5s 231ms/step - loss: 0.629
8 - accuracy: 0.7379 - val_loss: 0.6261 - val_accuracy: 0.7400
Epoch 82/100
21/21 [==============================] - 5s 233ms/step - loss: 0.558
2 - accuracy: 0.7621 - val_loss: 0.7064 - val_accuracy: 0.6600
Epoch 83/100
```

```
21/21 [==============================] - 5s 237ms/step - loss: 0.506
8 - accuracy: 0.7767 - val_loss: 0.4752 - val_accuracy: 0.7800
Epoch 84/100
21/21 [==============================] - 5s 224ms/step - loss: 0.502
9 - accuracy: 0.7573 - val_loss: 0.7053 - val_accuracy: 0.6600
Epoch 85/100
21/21 [==============================] - 5s 235ms/step - loss: 0.532
3 - accuracy: 0.7767 - val_loss: 0.7489 - val_accuracy: 0.6400
Epoch 86/100
21/21 [==============================] - 5s 235ms/step - loss: 0.476
6 - accuracy: 0.8010 - val_loss: 0.6302 - val_accuracy: 0.7200
Epoch 87/100
21/21 [==============================] - 5s 235ms/step - loss: 0.478
8 - accuracy: 0.8301 - val_loss: 0.6049 - val_accuracy: 0.7000
Epoch 88/100
21/21 [==============================] - 5s 236ms/step - loss: 0.487
1 - accuracy: 0.7816 - val_loss: 0.7066 - val_accuracy: 0.6600
Epoch 89/100
21/21 [==============================] - 5s 228ms/step - loss: 0.490
1 - accuracy: 0.7816 - val_loss: 0.7800 - val_accuracy: 0.6400
Epoch 90/100
21/21 [==============================] - 5s 238ms/step - loss: 0.434
3 - accuracy: 0.8350 - val_loss: 0.5082 - val_accuracy: 0.7400
Epoch 91/100
21/21 [==============================] - 5s 236ms/step - loss: 0.497
0 - accuracy: 0.7767 - val_loss: 0.4633 - val_accuracy: 0.7800
Epoch 92/100
21/21 [==============================] - 5s 231ms/step - loss: 0.501
9 - accuracy: 0.8058 - val_loss: 0.6147 - val_accuracy: 0.7000
Epoch 93/100
21/21 [==============================] - 5s 235ms/step - loss: 0.535
9 - accuracy: 0.7573 - val_loss: 0.7254 - val_accuracy: 0.6400
Epoch 94/100
21/21 [==============================] - 5s 233ms/step - loss: 0.441
3 - accuracy: 0.8010 - val_loss: 0.8067 - val_accuracy: 0.7000
Epoch 95/100
21/21 [==============================] - 5s 235ms/step - loss: 0.494
2 - accuracy: 0.7913 - val_loss: 0.6256 - val_accuracy: 0.7400
Epoch 96/100
21/21 [==============================] - 5s 234ms/step - loss: 0.453
7 - accuracy: 0.8010 - val_loss: 0.6329 - val_accuracy: 0.6600
Epoch 97/100
21/21 [==============================] - 5s 226ms/step - loss: 0.448
8 - accuracy: 0.8010 - val_loss: 0.7618 - val_accuracy: 0.6200
Epoch 98/100
21/21 [==============================] - 5s 234ms/step - loss: 0.462
1 - accuracy: 0.7816 - val_loss: 0.6498 - val_accuracy: 0.6800
Epoch 99/100
21/21 [==============================] - 5s 231ms/step - loss: 0.493
8 - accuracy: 0.7767 - val_loss: 0.8265 - val_accuracy: 0.6600
Epoch 100/100
21/21 [==============================] - 5s 227ms/step - loss: 0.471
8 - accuracy: 0.8058 - val_loss: 0.9059 - val_accuracy: 0.6800
```

[5 points] Plot Accuracy and Loss During Training

In [7]:

```
import matplotlib.pyplot as plt

def plot_acc_loss(history, epochs):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    plt.figure(figsize=(15, 5))

    plt.subplot(121)
    plt.plot(range(1,epochs), acc[1:], label='Train_acc')
    plt.plot(range(1,epochs), val_acc[1:], label='Test_acc')
    plt.title('Accuracy over ' + str(epochs) + ' Epochs', size=15)
    plt.grid(True)
    plt.legend()

    plt.subplot(122)
    plt.plot(range(1,epochs), loss[1:], label='Train_loss')
    plt.plot(range(1,epochs), val_loss[1:], label='Test_loss')
    plt.title('Loss over ' + str(epochs) + ' Epochs', size=15)
    plt.grid(True)
    plt.legend()

    plt.show()


plot_acc_loss(history, 100)
```



Testing Model

In [8]:

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

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZ
E,
                                                  batch_size=1,shuffle=True,seed
=42,class_mode="categorical")
eval_generator.reset()
print(len(eval_generator))
x = model.evaluate_generator(eval_generator,steps = np.ceil(len(eval_generator))
,
                             use_multiprocessing = False,verbose = 1,workers=1)
print('Test loss:' , x[0])
print('Test accuracy:',x[1])
```

```
Found 36 images belonging to 4 classes.
36

C:\Users\Li\anaconda3\lib\site-packages\keras\engine\training.py:200
6: UserWarning: `Model.evaluate_generator` is deprecated and will be
removed in a future version. Please use `Model.evaluate`, which supp
orts generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '

36/36 [==============================] - 1s 16ms/step - loss: 0.7485
- accuracy: 0.7500
Test loss: 0.748456597328186
Test accuracy: 0.75
```

[10 points] TSNE Plot

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

In [11]:

```
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.models.Model(inputs=model.input,
                                                 outputs=model.get_layer('dense').output)
tsne_eval_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=
IMAGE_SIZE,
                                                       batch_size=1,shuffle=False,see
d=42,class_mode="categorical")
print(tsne_eval_generator.class_indices)

outputs = intermediate_layer_model.predict_generator(tsne_eval_generator,270,ver
bose=1)
features = TSNE(n_components=2).fit_transform(outputs)
label = tsne_eval_generator.classes


covid_x, covid_y, normal_x, normal_y, pneumonia_bac_x, pneumonia_bac_y, pneumoni
a_vir_x, pneumonia_vir_y = [],[],[],[],[],[],[],[]

plt.figure()
for index in range(len(features)):
    if label[index] == 0:
        covid_x.append(features[index, 0])
        covid_y.append(features[index, 1])
    elif label[index] == 1:
        normal_x.append(features[index, 0])
        normal_y.append(features[index, 1])
    elif label[index] == 2:
        pneumonia_bac_x.append(features[index, 0])
        pneumonia_bac_y.append(features[index, 1])
    else:
        pneumonia_vir_x.append(features[index, 0])
        pneumonia_vir_y.append(features[index, 1])

plt.plot(covid_x, covid_y,'bo', ms=4, label="COVID-19")
plt.plot(normal_x, normal_y,'yo', ms=4, label="Normal")
plt.plot(pneumonia_bac_x, pneumonia_bac_y, 'go', ms=4, label="Pneumonia_bac")
plt.plot(pneumonia_vir_x, pneumonia_vir_y, 'ro', ms=4, label="Pneumonia_vir")

plt.legend(loc='upper left')
```

```
Found 270 images belonging to 4 classes.
{'covid': 0, 'normal': 1, 'pneumonia_bac': 2, 'pneumonia_vir': 3}
  1/270 [..............................] - ETA: 29s

C:\Users\Li\anaconda3\lib\site-packages\keras\engine\training.py:203
5: UserWarning: `Model.predict_generator` is deprecated and will be
removed in a future version. Please use `Model.predict`, which suppo
rts generators.
  warnings.warn('`Model.predict_generator` is deprecated and '

270/270 [==============================] - 4s 13ms/step
```

Out[11]:

```
<matplotlib.legend.Legend at 0x29993192bc8>
```