task2

December 4, 2020

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

2 Task 2 [Total points: 30]

2.1 Setup

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

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

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

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

2.2 Data

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

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

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

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

2.3 [20 points] Multi-class Classification

[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__
```

[1]: '2.3.1'

Load Image Data

[2]:
```
DATA_LIST = os.listdir('all/train')
DATASET_PATH  = 'all/train'
TEST_DIR =  'all/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 10  # try reducing batch size or freeze more layers if your GPU␣
 ↪runs out of memory
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experiment␣
 ↪with reducing it gradually
```

Generate Training and Validation Batches

[3]:
```
train_datagen = ImageDataGenerator(rescale=1./
 ↪255,rotation_range=50,featurewise_center = True,
                                   featurewise_std_normalization =␣
 ↪True,width_shift_range=0.2,
                                   height_shift_range=0.2,shear_range=0.
 ↪25,zoom_range=0.1,
                                   zca_whitening = True,channel_shift_range =␣
 ↪20,
                                   horizontal_flip = True,vertical_flip = True,
                                   validation_split = 0.2,fill_mode='constant')


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

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

```
/Users/bellarocha/anaconda3/envs/tf2/lib/python3.7/site-
packages/keras_preprocessing/image/image_data_generator.py:342: UserWarning:
This ImageDataGenerator specifies `zca_whitening` which overrides setting
of`featurewise_std_normalization`.
  warnings.warn('This ImageDataGenerator specifies '
```

```
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.

[4]:
```
vgg16 = tf.keras.applications.VGG16(
    include_top=False,
    weights="imagenet",
    input_shape= (224, 224, 3)
)

model = tf.keras.models.Sequential([
    vgg16,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation = 'relu', name = 'dense_feature'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(4, activation = 'softmax')
])

model.summary()
```

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

[5 points] Train Model

[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

OPTIMIZER = tf.keras.optimizers.SGD(learning_rate = LEARNING_RATE)

model.compile(optimizer = OPTIMIZER, loss = tf.keras.losses.
 ↪CategoricalCrossentropy(), metrics = ['accuracy'])

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

```
22
6

/Users/bellarocha/anaconda3/envs/tf2/lib/python3.7/site-
packages/keras_preprocessing/image/image_data_generator.py:720: UserWarning:
This ImageDataGenerator specifies `featurewise_center`, but it hasn't been fit
on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/Users/bellarocha/anaconda3/envs/tf2/lib/python3.7/site-
packages/keras_preprocessing/image/image_data_generator.py:739: UserWarning:
This ImageDataGenerator specifies `zca_whitening`, but it hasn't been fit on any
training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '

Epoch 1/100
21/21 [==============================] - 170s 8s/step - loss: 1.4831 - accuracy:
0.2379 - val_loss: 1.3861 - val_accuracy: 0.3200
Epoch 2/100
21/21 [==============================] - 176s 8s/step - loss: 1.4086 - accuracy:
0.2864 - val_loss: 1.3641 - val_accuracy: 0.3400
Epoch 3/100
21/21 [==============================] - 177s 8s/step - loss: 1.4000 - accuracy:
0.2952 - val_loss: 1.3520 - val_accuracy: 0.2800
Epoch 4/100
21/21 [==============================] - 176s 8s/step - loss: 1.4447 - accuracy:
0.2864 - val_loss: 1.4022 - val_accuracy: 0.2400
Epoch 5/100
21/21 [==============================] - 179s 9s/step - loss: 1.3986 - accuracy:
0.2913 - val_loss: 1.3295 - val_accuracy: 0.3000
Epoch 6/100
21/21 [==============================] - 178s 8s/step - loss: 1.3687 - accuracy:
0.3592 - val_loss: 1.3591 - val_accuracy: 0.2800
```

```
Epoch 7/100
21/21 [==============================] - 178s 8s/step - loss: 1.3365 - accuracy:
0.3641 - val_loss: 1.3229 - val_accuracy: 0.3200
Epoch 8/100
21/21 [==============================] - 174s 8s/step - loss: 1.3223 - accuracy:
0.3495 - val_loss: 1.3090 - val_accuracy: 0.3600
Epoch 9/100
21/21 [==============================] - 180s 9s/step - loss: 1.3357 - accuracy:
0.3592 - val_loss: 1.2514 - val_accuracy: 0.4400
Epoch 10/100
21/21 [==============================] - 187s 9s/step - loss: 1.2694 - accuracy:
0.4126 - val_loss: 1.2434 - val_accuracy: 0.4800
Epoch 11/100
21/21 [==============================] - 182s 9s/step - loss: 1.2990 - accuracy:
0.3883 - val_loss: 1.3538 - val_accuracy: 0.3600
Epoch 12/100
21/21 [==============================] - 175s 8s/step - loss: 1.3207 - accuracy:
0.4078 - val_loss: 1.2622 - val_accuracy: 0.5200
Epoch 13/100
21/21 [==============================] - 216s 10s/step - loss: 1.2141 -
accuracy: 0.4854 - val_loss: 1.2075 - val_accuracy: 0.5200
Epoch 14/100
21/21 [==============================] - 219s 10s/step - loss: 1.2501 -
accuracy: 0.4417 - val_loss: 1.1969 - val_accuracy: 0.4600
Epoch 15/100
21/21 [==============================] - 220s 10s/step - loss: 1.1790 -
accuracy: 0.4903 - val_loss: 1.2188 - val_accuracy: 0.4200
Epoch 16/100
21/21 [==============================] - 202s 10s/step - loss: 1.2276 -
accuracy: 0.4369 - val_loss: 1.1616 - val_accuracy: 0.4800
Epoch 17/100
21/21 [==============================] - 202s 10s/step - loss: 1.1636 -
accuracy: 0.5194 - val_loss: 1.1476 - val_accuracy: 0.5200
Epoch 18/100
21/21 [==============================] - 194s 9s/step - loss: 1.1814 - accuracy:
0.4563 - val_loss: 1.1449 - val_accuracy: 0.4400
Epoch 19/100
21/21 [==============================] - 189s 9s/step - loss: 1.1601 - accuracy:
0.4951 - val_loss: 1.1310 - val_accuracy: 0.4400
Epoch 20/100
21/21 [==============================] - 191s 9s/step - loss: 1.1826 - accuracy:
0.4563 - val_loss: 1.1608 - val_accuracy: 0.3800
Epoch 21/100
21/21 [==============================] - 191s 9s/step - loss: 1.0885 - accuracy:
0.5097 - val_loss: 1.0241 - val_accuracy: 0.5200
Epoch 22/100
21/21 [==============================] - 186s 9s/step - loss: 1.1017 - accuracy:
0.5000 - val_loss: 1.1085 - val_accuracy: 0.4600
```

```
Epoch 23/100
21/21 [==============================] - 183s 9s/step - loss: 1.0844 - accuracy:
0.5437 - val_loss: 1.0574 - val_accuracy: 0.5400
Epoch 24/100
21/21 [==============================] - 183s 9s/step - loss: 1.0671 - accuracy:
0.5583 - val_loss: 1.1433 - val_accuracy: 0.4400
Epoch 25/100
21/21 [==============================] - 183s 9s/step - loss: 1.0619 - accuracy:
0.5243 - val_loss: 0.9973 - val_accuracy: 0.5000
Epoch 26/100
21/21 [==============================] - 186s 9s/step - loss: 1.0109 - accuracy:
0.5534 - val_loss: 0.9349 - val_accuracy: 0.6800
Epoch 27/100
21/21 [==============================] - 185s 9s/step - loss: 1.0164 - accuracy:
0.5340 - val_loss: 1.0014 - val_accuracy: 0.4600
Epoch 28/100
21/21 [==============================] - 185s 9s/step - loss: 1.0460 - accuracy:
0.5388 - val_loss: 1.0006 - val_accuracy: 0.5800
Epoch 29/100
21/21 [==============================] - 190s 9s/step - loss: 0.9979 - accuracy:
0.5146 - val_loss: 0.9653 - val_accuracy: 0.5600
Epoch 30/100
21/21 [==============================] - 155s 7s/step - loss: 1.0037 - accuracy:
0.5680 - val_loss: 1.0347 - val_accuracy: 0.4400
Epoch 31/100
21/21 [==============================] - 162s 8s/step - loss: 0.9682 - accuracy:
0.5874 - val_loss: 0.9815 - val_accuracy: 0.5600
Epoch 32/100
21/21 [==============================] - 167s 8s/step - loss: 0.9538 - accuracy:
0.5728 - val_loss: 0.9562 - val_accuracy: 0.5400
Epoch 33/100
21/21 [==============================] - 166s 8s/step - loss: 0.9485 - accuracy:
0.6117 - val_loss: 0.9852 - val_accuracy: 0.6000
Epoch 34/100
21/21 [==============================] - 168s 8s/step - loss: 0.9750 - accuracy:
0.5583 - val_loss: 0.9021 - val_accuracy: 0.6200
Epoch 35/100
21/21 [==============================] - 210s 10s/step - loss: 0.9397 -
accuracy: 0.6068 - val_loss: 0.9392 - val_accuracy: 0.6000
Epoch 36/100
21/21 [==============================] - 190s 9s/step - loss: 1.0140 - accuracy:
0.5583 - val_loss: 0.9210 - val_accuracy: 0.6200
Epoch 37/100
21/21 [==============================] - 155s 7s/step - loss: 0.9892 - accuracy:
0.6019 - val_loss: 0.9046 - val_accuracy: 0.5400
Epoch 38/100
21/21 [==============================] - 155s 7s/step - loss: 1.0141 - accuracy:
0.5534 - val_loss: 0.8966 - val_accuracy: 0.5800
```

```
Epoch 39/100
21/21 [==============================] - 180s 9s/step - loss: 0.9033 - accuracy:
0.6311 - val_loss: 0.9538 - val_accuracy: 0.5000
Epoch 40/100
21/21 [==============================] - 182s 9s/step - loss: 0.9211 - accuracy:
0.5728 - val_loss: 0.8999 - val_accuracy: 0.5400
Epoch 41/100
21/21 [==============================] - 178s 8s/step - loss: 0.9281 - accuracy:
0.5874 - val_loss: 0.9470 - val_accuracy: 0.5000
Epoch 42/100
21/21 [==============================] - 159s 8s/step - loss: 0.9226 - accuracy:
0.6068 - val_loss: 0.8460 - val_accuracy: 0.6200
Epoch 43/100
21/21 [==============================] - 155s 7s/step - loss: 0.9154 - accuracy:
0.6019 - val_loss: 0.8944 - val_accuracy: 0.6400
Epoch 44/100
21/21 [==============================] - 154s 7s/step - loss: 0.9156 - accuracy:
0.6068 - val_loss: 0.8245 - val_accuracy: 0.6200
Epoch 45/100
21/21 [==============================] - 152s 7s/step - loss: 0.8920 - accuracy:
0.6117 - val_loss: 0.9446 - val_accuracy: 0.5400
Epoch 46/100
21/21 [==============================] - 151s 7s/step - loss: 0.8794 - accuracy:
0.6262 - val_loss: 0.9039 - val_accuracy: 0.5800
Epoch 47/100
21/21 [==============================] - 155s 7s/step - loss: 0.8761 - accuracy:
0.6359 - val_loss: 0.8983 - val_accuracy: 0.5800
Epoch 48/100
21/21 [==============================] - 151s 7s/step - loss: 0.8763 - accuracy:
0.6214 - val_loss: 0.9053 - val_accuracy: 0.5600
Epoch 49/100
21/21 [==============================] - 160s 8s/step - loss: 0.8953 - accuracy:
0.6165 - val_loss: 0.7793 - val_accuracy: 0.6600
Epoch 50/100
21/21 [==============================] - 155s 7s/step - loss: 0.8588 - accuracy:
0.6359 - val_loss: 0.8608 - val_accuracy: 0.6200
Epoch 51/100
21/21 [==============================] - 152s 7s/step - loss: 0.8890 - accuracy:
0.6117 - val_loss: 0.8491 - val_accuracy: 0.5600
Epoch 52/100
21/21 [==============================] - 152s 7s/step - loss: 0.8599 - accuracy:
0.6408 - val_loss: 0.8872 - val_accuracy: 0.6200
Epoch 53/100
21/21 [==============================] - 151s 7s/step - loss: 0.8348 - accuracy:
0.6553 - val_loss: 0.9573 - val_accuracy: 0.5200
Epoch 54/100
21/21 [==============================] - 153s 7s/step - loss: 0.8589 - accuracy:
0.6311 - val_loss: 0.8699 - val_accuracy: 0.6000
```

```
Epoch 55/100
21/21 [==============================] - 149s 7s/step - loss: 0.8641 - accuracy:
0.6019 - val_loss: 0.8377 - val_accuracy: 0.6000
Epoch 56/100
21/21 [==============================] - 159s 8s/step - loss: 0.8258 - accuracy:
0.6699 - val_loss: 0.7211 - val_accuracy: 0.6400
Epoch 57/100
21/21 [==============================] - 184s 9s/step - loss: 0.8092 - accuracy:
0.6456 - val_loss: 0.7379 - val_accuracy: 0.6200
Epoch 58/100
21/21 [==============================] - 235s 11s/step - loss: 0.8111 -
accuracy: 0.6165 - val_loss: 0.8134 - val_accuracy: 0.6400
Epoch 59/100
21/21 [==============================] - 223s 11s/step - loss: 0.7904 -
accuracy: 0.6796 - val_loss: 0.8412 - val_accuracy: 0.5600
Epoch 60/100
21/21 [==============================] - 224s 11s/step - loss: 0.8776 -
accuracy: 0.6262 - val_loss: 0.7700 - val_accuracy: 0.6400
Epoch 61/100
21/21 [==============================] - 192s 9s/step - loss: 0.8060 - accuracy:
0.6602 - val_loss: 0.8034 - val_accuracy: 0.6200
Epoch 62/100
21/21 [==============================] - 187s 9s/step - loss: 0.8465 - accuracy:
0.6068 - val_loss: 0.8220 - val_accuracy: 0.5400
Epoch 63/100
21/21 [==============================] - 187s 9s/step - loss: 0.8612 - accuracy:
0.6311 - val_loss: 0.8547 - val_accuracy: 0.5600
Epoch 64/100
21/21 [==============================] - 189s 9s/step - loss: 0.7903 - accuracy:
0.7184 - val_loss: 0.8515 - val_accuracy: 0.6400
Epoch 65/100
21/21 [==============================] - 186s 9s/step - loss: 0.8276 - accuracy:
0.6408 - val_loss: 0.7937 - val_accuracy: 0.6400
Epoch 66/100
21/21 [==============================] - 184s 9s/step - loss: 0.8127 - accuracy:
0.6117 - val_loss: 0.8569 - val_accuracy: 0.6000
Epoch 67/100
21/21 [==============================] - 185s 9s/step - loss: 0.7931 - accuracy:
0.6699 - val_loss: 0.8449 - val_accuracy: 0.6200
Epoch 68/100
21/21 [==============================] - 184s 9s/step - loss: 0.8241 - accuracy:
0.6408 - val_loss: 0.6806 - val_accuracy: 0.7000
Epoch 69/100
21/21 [==============================] - 184s 9s/step - loss: 0.7606 - accuracy:
0.6845 - val_loss: 0.9621 - val_accuracy: 0.5600
Epoch 70/100
21/21 [==============================] - 183s 9s/step - loss: 0.8560 - accuracy:
0.5922 - val_loss: 0.8160 - val_accuracy: 0.5800
```

```
Epoch 71/100
21/21 [==============================] - 188s 9s/step - loss: 0.7296 - accuracy:
0.6699 - val_loss: 0.7795 - val_accuracy: 0.6200
Epoch 72/100
21/21 [==============================] - 184s 9s/step - loss: 0.8144 - accuracy:
0.6796 - val_loss: 0.8258 - val_accuracy: 0.5200
Epoch 73/100
21/21 [==============================] - 183s 9s/step - loss: 0.7676 - accuracy:
0.6505 - val_loss: 0.7192 - val_accuracy: 0.6800
Epoch 74/100
21/21 [==============================] - 187s 9s/step - loss: 0.7636 - accuracy:
0.6796 - val_loss: 0.8150 - val_accuracy: 0.6400
Epoch 75/100
21/21 [==============================] - 186s 9s/step - loss: 0.7654 - accuracy:
0.6893 - val_loss: 0.8493 - val_accuracy: 0.5400
Epoch 76/100
21/21 [==============================] - 188s 9s/step - loss: 0.7694 - accuracy:
0.6505 - val_loss: 0.6741 - val_accuracy: 0.6400
Epoch 77/100
21/21 [==============================] - 170s 8s/step - loss: 0.8262 - accuracy:
0.6262 - val_loss: 0.7648 - val_accuracy: 0.6400
Epoch 78/100
21/21 [==============================] - 150s 7s/step - loss: 0.8148 - accuracy:
0.6311 - val_loss: 0.7190 - val_accuracy: 0.6000
Epoch 79/100
21/21 [==============================] - 149s 7s/step - loss: 0.7550 - accuracy:
0.6311 - val_loss: 0.6470 - val_accuracy: 0.7400
Epoch 80/100
21/21 [==============================] - 151s 7s/step - loss: 0.7279 - accuracy:
0.6942 - val_loss: 0.7523 - val_accuracy: 0.5800
Epoch 81/100
21/21 [==============================] - 153s 7s/step - loss: 0.7594 - accuracy:
0.6602 - val_loss: 0.6954 - val_accuracy: 0.6800
Epoch 82/100
21/21 [==============================] - 153s 7s/step - loss: 0.7130 - accuracy:
0.6650 - val_loss: 0.7724 - val_accuracy: 0.5800
Epoch 83/100
21/21 [==============================] - 151s 7s/step - loss: 0.7335 - accuracy:
0.6893 - val_loss: 0.6850 - val_accuracy: 0.6000
Epoch 84/100
21/21 [==============================] - 152s 7s/step - loss: 0.7554 - accuracy:
0.6456 - val_loss: 0.7766 - val_accuracy: 0.6400
Epoch 85/100
21/21 [==============================] - 153s 7s/step - loss: 0.7117 - accuracy:
0.7039 - val_loss: 0.7066 - val_accuracy: 0.7000
Epoch 86/100
21/21 [==============================] - 149s 7s/step - loss: 0.7286 - accuracy:
0.6845 - val_loss: 0.8997 - val_accuracy: 0.5600
```

```
Epoch 87/100
21/21 [==============================] - 150s 7s/step - loss: 0.7224 - accuracy:
0.6990 - val_loss: 0.7625 - val_accuracy: 0.6200
Epoch 88/100
21/21 [==============================] - 149s 7s/step - loss: 0.7321 - accuracy:
0.6699 - val_loss: 0.6604 - val_accuracy: 0.6400
Epoch 89/100
21/21 [==============================] - 151s 7s/step - loss: 0.7194 - accuracy:
0.6553 - val_loss: 0.8738 - val_accuracy: 0.6200
Epoch 90/100
21/21 [==============================] - 154s 7s/step - loss: 0.6424 - accuracy:
0.7039 - val_loss: 0.7508 - val_accuracy: 0.6200
Epoch 91/100
21/21 [==============================] - 150s 7s/step - loss: 0.7899 - accuracy:
0.6505 - val_loss: 0.7488 - val_accuracy: 0.6000
Epoch 92/100
21/21 [==============================] - 150s 7s/step - loss: 0.7175 - accuracy:
0.6845 - val_loss: 0.7415 - val_accuracy: 0.6600
Epoch 93/100
21/21 [==============================] - 153s 7s/step - loss: 0.7343 - accuracy:
0.7087 - val_loss: 0.7054 - val_accuracy: 0.6800
Epoch 94/100
21/21 [==============================] - 150s 7s/step - loss: 0.7121 - accuracy:
0.6650 - val_loss: 0.6860 - val_accuracy: 0.5600
Epoch 95/100
21/21 [==============================] - 151s 7s/step - loss: 0.7031 - accuracy:
0.6699 - val_loss: 0.8536 - val_accuracy: 0.6000
Epoch 96/100
21/21 [==============================] - 153s 7s/step - loss: 0.7146 - accuracy:
0.6714 - val_loss: 0.7106 - val_accuracy: 0.7000
Epoch 97/100
21/21 [==============================] - 150s 7s/step - loss: 0.6925 - accuracy:
0.6942 - val_loss: 0.7039 - val_accuracy: 0.6600
Epoch 98/100
21/21 [==============================] - 152s 7s/step - loss: 0.6457 - accuracy:
0.7233 - val_loss: 0.6853 - val_accuracy: 0.6600
Epoch 99/100
21/21 [==============================] - 150s 7s/step - loss: 0.6676 - accuracy:
0.7379 - val_loss: 0.7534 - val_accuracy: 0.6600
Epoch 100/100
21/21 [==============================] - 152s 7s/step - loss: 0.6589 - accuracy:
0.7379 - val_loss: 0.7270 - val_accuracy: 0.6400
```

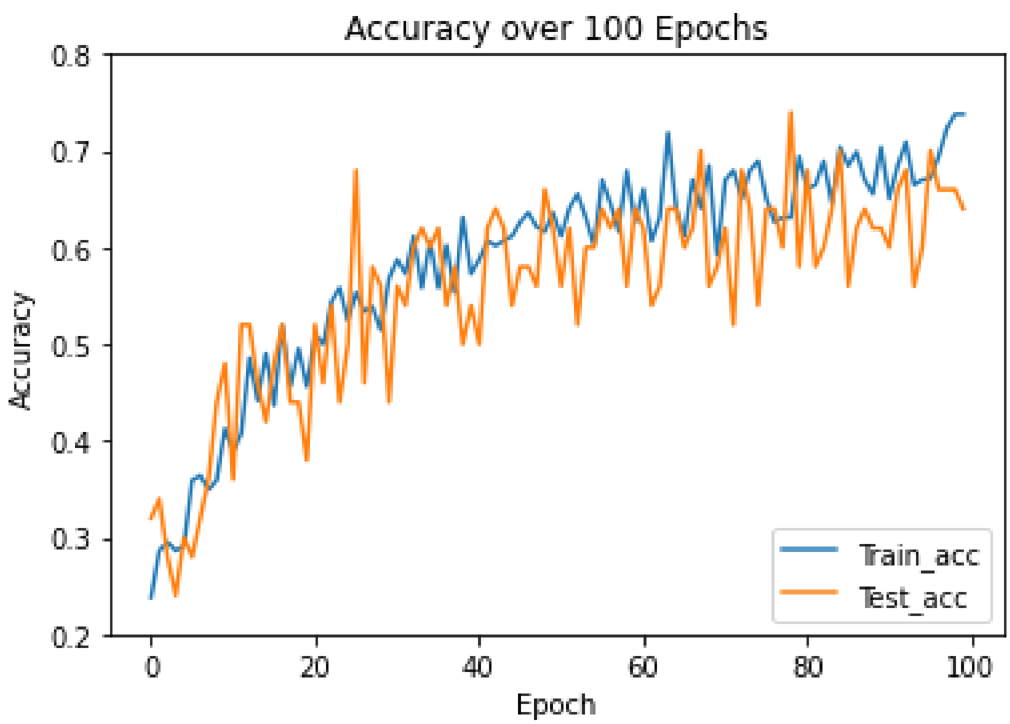
[5 points] Plot Accuracy and Loss During Training

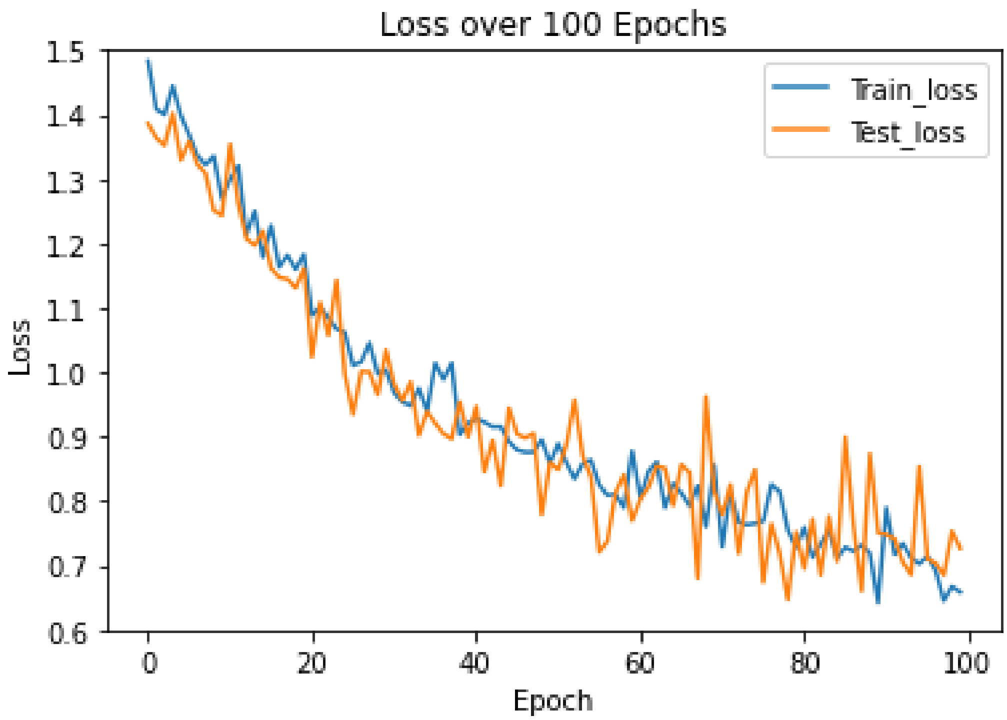
[22]:
```python
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Train_acc')
```

```
plt.plot(history.history['val_accuracy'], label = 'Test_acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.2, 0.8])
plt.legend(loc='lower right')
plt.title('Accuracy over 100 Epochs')
plt.show()

plt.plot(history.history['loss'], label='Train_loss')
plt.plot(history.history['val_loss'], label = 'Test_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.ylim([0.6, 1.5])
plt.legend(loc='upper right')
plt.title('Loss over 100 Epochs')
plt.show()
```





Testing Model

[28]:
```
test_datagen = ImageDataGenerator(rescale=1. / 255)

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

```
Found 36 images belonging to 4 classes.
36
36/36 [==============================] - 17s 466ms/step - loss: 1.3225 -
accuracy: 0.3611
Test loss: 1.3225327730178833
Test accuracy: 0.3611111044883728
```

2.4 [10 points] TSNE Plot

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

[16]:
```
from sklearn.manifold import TSNE

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_SIZE,
                                                      ␣
 ↪batch_size=1,shuffle=False,seed=42,class_mode="categorical")

intermediate_layer = intermediate_layer_model.predict(tsne_eval_generator)

intermediate_layer_TSNE = TSNE().fit_transform(intermediate_layer)

classes = tsne_eval_generator.classes

colors = []

for i in range(len(classes)):
    if classes[i] == 0:
        colors.append('blue')
    if classes[i] == 1:
        colors.append('orange')
    if classes[i] == 2:
        colors.append('green')
    if classes[i] == 3:
        colors.append('red')

plt.scatter(intermediate_layer_TSNE[:, 0], intermediate_layer_TSNE[:, 1], color␣
 ↪= colors)
plt.scatter(intermediate_layer_TSNE[:, 0][0], intermediate_layer_TSNE[:, 1][0],␣
 ↪color = colors[0], label = 'COVID-19')
plt.scatter(intermediate_layer_TSNE[:, 0][70], intermediate_layer_TSNE[:,␣
 ↪1][70], color = colors[70], label = 'Normal')
plt.scatter(intermediate_layer_TSNE[:, 0][170], intermediate_layer_TSNE[:,␣
 ↪1][70], color = colors[170], label = 'Pneumonia_bac')
plt.scatter(intermediate_layer_TSNE[:, 0][250], intermediate_layer_TSNE[:,␣
 ↪1][250], color = colors[250], label = 'Pneumonia_vir')
plt.legend()
```

```
plt.show()
```

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