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/benreichelt/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(weights="imagenet", include_top=False,␣
↪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(0.3),
  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

[5]:
```
#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

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

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

#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, validation_data = valid_batches,
 ↪validation_steps=STEP_SIZE_VALID)
```

```
22
6
```

```
/Users/benreichelt/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/benreichelt/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 [==============================] - 70s 3s/step - loss: 1.5843 - accuracy:
0.2039 - val_loss: 1.3858 - val_accuracy: 0.3200
Epoch 2/100
21/21 [==============================] - 79s 4s/step - loss: 1.4542 - accuracy:
0.2961 - val_loss: 1.3780 - val_accuracy: 0.3200
Epoch 3/100
21/21 [==============================] - 73s 3s/step - loss: 1.4205 - accuracy:
0.3010 - val_loss: 1.3735 - val_accuracy: 0.3000
Epoch 4/100
21/21 [==============================] - 74s 4s/step - loss: 1.4723 - accuracy:
0.3058 - val_loss: 1.3648 - val_accuracy: 0.2200
Epoch 5/100
21/21 [==============================] - 74s 4s/step - loss: 1.4189 - accuracy:
0.2621 - val_loss: 1.3308 - val_accuracy: 0.4800
Epoch 6/100
21/21 [==============================] - 72s 3s/step - loss: 1.3896 - accuracy:
0.2961 - val_loss: 1.3303 - val_accuracy: 0.3200
Epoch 7/100
21/21 [==============================] - 71s 3s/step - loss: 1.3759 - accuracy:
```

```
0.3689 - val_loss: 1.3468 - val_accuracy: 0.4000
Epoch 8/100
21/21 [==============================] - 71s 3s/step - loss: 1.3832 - accuracy:
0.3155 - val_loss: 1.3177 - val_accuracy: 0.3600
Epoch 9/100
21/21 [==============================] - 71s 3s/step - loss: 1.3102 - accuracy:
0.4078 - val_loss: 1.3063 - val_accuracy: 0.3400
Epoch 10/100
21/21 [==============================] - 70s 3s/step - loss: 1.3246 - accuracy:
0.3786 - val_loss: 1.2503 - val_accuracy: 0.4800
Epoch 11/100
21/21 [==============================] - 71s 3s/step - loss: 1.3184 - accuracy:
0.3932 - val_loss: 1.2619 - val_accuracy: 0.3600
Epoch 12/100
21/21 [==============================] - 70s 3s/step - loss: 1.2696 - accuracy:
0.4320 - val_loss: 1.2373 - val_accuracy: 0.4000
Epoch 13/100
21/21 [==============================] - 70s 3s/step - loss: 1.2465 - accuracy:
0.4175 - val_loss: 1.2588 - val_accuracy: 0.3800
Epoch 14/100
21/21 [==============================] - 71s 3s/step - loss: 1.2464 - accuracy:
0.4466 - val_loss: 1.1567 - val_accuracy: 0.5800
Epoch 15/100
21/21 [==============================] - 70s 3s/step - loss: 1.2545 - accuracy:
0.4126 - val_loss: 1.1790 - val_accuracy: 0.5000
Epoch 16/100
21/21 [==============================] - 70s 3s/step - loss: 1.2141 - accuracy:
0.4272 - val_loss: 1.1313 - val_accuracy: 0.4000
Epoch 17/100
21/21 [==============================] - 70s 3s/step - loss: 1.1959 - accuracy:
0.4612 - val_loss: 1.0900 - val_accuracy: 0.5800
Epoch 18/100
21/21 [==============================] - 70s 3s/step - loss: 1.1882 - accuracy:
0.4612 - val_loss: 1.1424 - val_accuracy: 0.5000
Epoch 19/100
21/21 [==============================] - 70s 3s/step - loss: 1.1972 - accuracy:
0.4272 - val_loss: 1.0727 - val_accuracy: 0.4800
Epoch 20/100
21/21 [==============================] - 70s 3s/step - loss: 1.1594 - accuracy:
0.5000 - val_loss: 1.1504 - val_accuracy: 0.4600
Epoch 21/100
21/21 [==============================] - 70s 3s/step - loss: 1.1646 - accuracy:
0.4417 - val_loss: 1.0198 - val_accuracy: 0.4800
Epoch 22/100
21/21 [==============================] - 71s 3s/step - loss: 1.1595 - accuracy:
0.4563 - val_loss: 1.0715 - val_accuracy: 0.5600
Epoch 23/100
21/21 [==============================] - 85s 4s/step - loss: 1.0783 - accuracy:
```

```
0.5146 - val_loss: 1.0301 - val_accuracy: 0.5200
Epoch 24/100
21/21 [==============================] - 70s 3s/step - loss: 1.0596 - accuracy:
0.5291 - val_loss: 1.0478 - val_accuracy: 0.5600
Epoch 25/100
21/21 [==============================] - 72s 3s/step - loss: 1.0532 - accuracy:
0.5048 - val_loss: 0.9626 - val_accuracy: 0.4800
Epoch 26/100
21/21 [==============================] - 71s 3s/step - loss: 1.0753 - accuracy:
0.5194 - val_loss: 0.9132 - val_accuracy: 0.6200
Epoch 27/100
21/21 [==============================] - 70s 3s/step - loss: 1.0544 - accuracy:
0.5340 - val_loss: 0.9824 - val_accuracy: 0.5400
Epoch 28/100
21/21 [==============================] - 71s 3s/step - loss: 1.0605 - accuracy:
0.5340 - val_loss: 0.9590 - val_accuracy: 0.5200
Epoch 29/100
21/21 [==============================] - 70s 3s/step - loss: 1.0423 - accuracy:
0.5291 - val_loss: 0.9719 - val_accuracy: 0.5800
Epoch 30/100
21/21 [==============================] - 70s 3s/step - loss: 1.0108 - accuracy:
0.5825 - val_loss: 0.9476 - val_accuracy: 0.5600
Epoch 31/100
21/21 [==============================] - 71s 3s/step - loss: 1.0282 - accuracy:
0.5762 - val_loss: 0.9218 - val_accuracy: 0.5800
Epoch 32/100
21/21 [==============================] - 70s 3s/step - loss: 1.0351 - accuracy:
0.5097 - val_loss: 0.9387 - val_accuracy: 0.5600
Epoch 33/100
21/21 [==============================] - 76s 4s/step - loss: 0.9644 - accuracy:
0.6165 - val_loss: 0.9321 - val_accuracy: 0.6400
Epoch 34/100
21/21 [==============================] - 81s 4s/step - loss: 0.9920 - accuracy:
0.5388 - val_loss: 0.8970 - val_accuracy: 0.5800
Epoch 35/100
21/21 [==============================] - 72s 3s/step - loss: 0.9950 - accuracy:
0.5388 - val_loss: 0.8761 - val_accuracy: 0.5400
Epoch 36/100
21/21 [==============================] - 70s 3s/step - loss: 0.9782 - accuracy:
0.5583 - val_loss: 0.9011 - val_accuracy: 0.5400
Epoch 37/100
21/21 [==============================] - 70s 3s/step - loss: 0.9407 - accuracy:
0.5680 - val_loss: 0.8992 - val_accuracy: 0.5400
Epoch 38/100
21/21 [==============================] - 70s 3s/step - loss: 0.9641 - accuracy:
0.5437 - val_loss: 0.8883 - val_accuracy: 0.5600
Epoch 39/100
21/21 [==============================] - 70s 3s/step - loss: 0.9520 - accuracy:
```

```
0.6117 - val_loss: 0.8485 - val_accuracy: 0.5600
Epoch 40/100
21/21 [==============================] - 70s 3s/step - loss: 0.9061 - accuracy:
0.6262 - val_loss: 0.8440 - val_accuracy: 0.6600
Epoch 41/100
21/21 [==============================] - 70s 3s/step - loss: 0.9029 - accuracy:
0.6068 - val_loss: 0.9396 - val_accuracy: 0.5400
Epoch 42/100
21/21 [==============================] - 70s 3s/step - loss: 0.9049 - accuracy:
0.6214 - val_loss: 0.8223 - val_accuracy: 0.6600
Epoch 43/100
21/21 [==============================] - 70s 3s/step - loss: 0.9440 - accuracy:
0.5680 - val_loss: 0.8762 - val_accuracy: 0.5800
Epoch 44/100
21/21 [==============================] - 70s 3s/step - loss: 0.8932 - accuracy:
0.6019 - val_loss: 0.8354 - val_accuracy: 0.6400
Epoch 45/100
21/21 [==============================] - 72s 3s/step - loss: 0.8784 - accuracy:
0.6311 - val_loss: 0.9027 - val_accuracy: 0.5400
Epoch 46/100
21/21 [==============================] - 70s 3s/step - loss: 0.8696 - accuracy:
0.5583 - val_loss: 0.8261 - val_accuracy: 0.5600
Epoch 47/100
21/21 [==============================] - 70s 3s/step - loss: 0.8773 - accuracy:
0.6019 - val_loss: 0.8130 - val_accuracy: 0.5600
Epoch 48/100
21/21 [==============================] - 72s 3s/step - loss: 0.8950 - accuracy:
0.6262 - val_loss: 0.9635 - val_accuracy: 0.5600
Epoch 49/100
21/21 [==============================] - 73s 3s/step - loss: 0.8352 - accuracy:
0.6262 - val_loss: 0.7738 - val_accuracy: 0.6400
Epoch 50/100
21/21 [==============================] - 70s 3s/step - loss: 0.8920 - accuracy:
0.6262 - val_loss: 0.8752 - val_accuracy: 0.5400
Epoch 51/100
21/21 [==============================] - 71s 3s/step - loss: 0.8932 - accuracy:
0.5905 - val_loss: 0.8626 - val_accuracy: 0.5800
Epoch 52/100
21/21 [==============================] - 70s 3s/step - loss: 0.7782 - accuracy:
0.6602 - val_loss: 0.7880 - val_accuracy: 0.5800
Epoch 53/100
21/21 [==============================] - 71s 3s/step - loss: 0.8299 - accuracy:
0.6714 - val_loss: 0.8929 - val_accuracy: 0.5800
Epoch 54/100
21/21 [==============================] - 69s 3s/step - loss: 0.8765 - accuracy:
0.5631 - val_loss: 0.8079 - val_accuracy: 0.6000
Epoch 55/100
21/21 [==============================] - 69s 3s/step - loss: 0.8725 - accuracy:
```

```
0.6214 - val_loss: 0.7694 - val_accuracy: 0.6200
Epoch 56/100
21/21 [==============================] - 70s 3s/step - loss: 0.8370 - accuracy:
0.6286 - val_loss: 0.6710 - val_accuracy: 0.6800
Epoch 57/100
21/21 [==============================] - 68s 3s/step - loss: 0.8345 - accuracy:
0.6117 - val_loss: 0.7578 - val_accuracy: 0.6200
Epoch 58/100
21/21 [==============================] - 70s 3s/step - loss: 0.8118 - accuracy:
0.6165 - val_loss: 0.7787 - val_accuracy: 0.6600
Epoch 59/100
21/21 [==============================] - 70s 3s/step - loss: 0.8609 - accuracy:
0.6311 - val_loss: 0.8657 - val_accuracy: 0.6000
Epoch 60/100
21/21 [==============================] - 71s 3s/step - loss: 0.7792 - accuracy:
0.6748 - val_loss: 0.7734 - val_accuracy: 0.6000
Epoch 61/100
21/21 [==============================] - 70s 3s/step - loss: 0.8781 - accuracy:
0.6019 - val_loss: 0.7220 - val_accuracy: 0.6600
Epoch 62/100
21/21 [==============================] - 68s 3s/step - loss: 0.8086 - accuracy:
0.6359 - val_loss: 0.8538 - val_accuracy: 0.5400
Epoch 63/100
21/21 [==============================] - 69s 3s/step - loss: 0.8211 - accuracy:
0.5874 - val_loss: 0.7102 - val_accuracy: 0.6400
Epoch 64/100
21/21 [==============================] - 68s 3s/step - loss: 0.8377 - accuracy:
0.6311 - val_loss: 0.7680 - val_accuracy: 0.6000
Epoch 65/100
21/21 [==============================] - 68s 3s/step - loss: 0.8056 - accuracy:
0.6602 - val_loss: 0.6730 - val_accuracy: 0.7400
Epoch 66/100
21/21 [==============================] - 68s 3s/step - loss: 0.7715 - accuracy:
0.6893 - val_loss: 0.7282 - val_accuracy: 0.6400
Epoch 67/100
21/21 [==============================] - 69s 3s/step - loss: 0.7591 - accuracy:
0.6990 - val_loss: 0.8670 - val_accuracy: 0.5400
Epoch 68/100
21/21 [==============================] - 68s 3s/step - loss: 0.7750 - accuracy:
0.6990 - val_loss: 0.6892 - val_accuracy: 0.6200
Epoch 69/100
21/21 [==============================] - 68s 3s/step - loss: 0.8339 - accuracy:
0.6165 - val_loss: 0.7790 - val_accuracy: 0.6600
Epoch 70/100
21/21 [==============================] - 68s 3s/step - loss: 0.7931 - accuracy:
0.6359 - val_loss: 0.7802 - val_accuracy: 0.6600
Epoch 71/100
21/21 [==============================] - 68s 3s/step - loss: 0.7553 - accuracy:
```

```
0.6990 - val_loss: 0.7135 - val_accuracy: 0.6400
Epoch 72/100
21/21 [==============================] - 68s 3s/step - loss: 0.7959 - accuracy:
0.6748 - val_loss: 0.7632 - val_accuracy: 0.5600
Epoch 73/100
21/21 [==============================] - 68s 3s/step - loss: 0.7654 - accuracy:
0.6602 - val_loss: 0.6575 - val_accuracy: 0.7200
Epoch 74/100
21/21 [==============================] - 68s 3s/step - loss: 0.7429 - accuracy:
0.7136 - val_loss: 0.8125 - val_accuracy: 0.5600
Epoch 75/100
21/21 [==============================] - 68s 3s/step - loss: 0.7777 - accuracy:
0.6893 - val_loss: 0.7188 - val_accuracy: 0.6800
Epoch 76/100
21/21 [==============================] - 68s 3s/step - loss: 0.8000 - accuracy:
0.6408 - val_loss: 0.7395 - val_accuracy: 0.5800
Epoch 77/100
21/21 [==============================] - 69s 3s/step - loss: 0.7434 - accuracy:
0.6942 - val_loss: 0.6708 - val_accuracy: 0.7000
Epoch 78/100
21/21 [==============================] - 68s 3s/step - loss: 0.7348 - accuracy:
0.7039 - val_loss: 0.6504 - val_accuracy: 0.7400
Epoch 79/100
21/21 [==============================] - 69s 3s/step - loss: 0.7853 - accuracy:
0.6262 - val_loss: 0.6619 - val_accuracy: 0.7000
Epoch 80/100
21/21 [==============================] - 68s 3s/step - loss: 0.7791 - accuracy:
0.6748 - val_loss: 0.7038 - val_accuracy: 0.6200
Epoch 81/100
21/21 [==============================] - 69s 3s/step - loss: 0.7326 - accuracy:
0.6796 - val_loss: 0.6460 - val_accuracy: 0.7200
Epoch 82/100
21/21 [==============================] - 69s 3s/step - loss: 0.7468 - accuracy:
0.6408 - val_loss: 0.7036 - val_accuracy: 0.6600
Epoch 83/100
21/21 [==============================] - 69s 3s/step - loss: 0.7558 - accuracy:
0.6748 - val_loss: 0.6503 - val_accuracy: 0.6000
Epoch 84/100
21/21 [==============================] - 69s 3s/step - loss: 0.7619 - accuracy:
0.6602 - val_loss: 0.7274 - val_accuracy: 0.6400
Epoch 85/100
21/21 [==============================] - 68s 3s/step - loss: 0.6778 - accuracy:
0.7136 - val_loss: 0.6661 - val_accuracy: 0.6400
Epoch 86/100
21/21 [==============================] - 69s 3s/step - loss: 0.7280 - accuracy:
0.6942 - val_loss: 0.7059 - val_accuracy: 0.6800
Epoch 87/100
21/21 [==============================] - 69s 3s/step - loss: 0.7296 - accuracy:
```

```
0.6699 - val_loss: 0.7610 - val_accuracy: 0.5800
Epoch 88/100
21/21 [==============================] - 69s 3s/step - loss: 0.7379 - accuracy:
0.7282 - val_loss: 0.6957 - val_accuracy: 0.6400
Epoch 89/100
21/21 [==============================] - 69s 3s/step - loss: 0.7610 - accuracy:
0.6505 - val_loss: 0.6615 - val_accuracy: 0.6800
Epoch 90/100
21/21 [==============================] - 69s 3s/step - loss: 0.6738 - accuracy:
0.7282 - val_loss: 0.6427 - val_accuracy: 0.6600
Epoch 91/100
21/21 [==============================] - 68s 3s/step - loss: 0.7333 - accuracy:
0.6990 - val_loss: 0.6849 - val_accuracy: 0.6400
Epoch 92/100
21/21 [==============================] - 68s 3s/step - loss: 0.7367 - accuracy:
0.6893 - val_loss: 0.7250 - val_accuracy: 0.6000
Epoch 93/100
21/21 [==============================] - 68s 3s/step - loss: 0.6670 - accuracy:
0.7039 - val_loss: 0.7145 - val_accuracy: 0.6400
Epoch 94/100
21/21 [==============================] - 69s 3s/step - loss: 0.6342 - accuracy:
0.7330 - val_loss: 0.6711 - val_accuracy: 0.6400
Epoch 95/100
21/21 [==============================] - 69s 3s/step - loss: 0.7286 - accuracy:
0.6748 - val_loss: 0.7099 - val_accuracy: 0.6200
Epoch 96/100
21/21 [==============================] - 70s 3s/step - loss: 0.6959 - accuracy:
0.6942 - val_loss: 0.6275 - val_accuracy: 0.7400
Epoch 97/100
21/21 [==============================] - 68s 3s/step - loss: 0.6876 - accuracy:
0.6796 - val_loss: 0.5714 - val_accuracy: 0.7600
Epoch 98/100
21/21 [==============================] - 68s 3s/step - loss: 0.6956 - accuracy:
0.6748 - val_loss: 0.6753 - val_accuracy: 0.6400
Epoch 99/100
21/21 [==============================] - 71s 3s/step - loss: 0.7050 - accuracy:
0.6990 - val_loss: 0.7110 - val_accuracy: 0.6000
Epoch 100/100
21/21 [==============================] - 69s 3s/step - loss: 0.7683 - accuracy:
0.6505 - val_loss: 0.7464 - val_accuracy: 0.6400
```

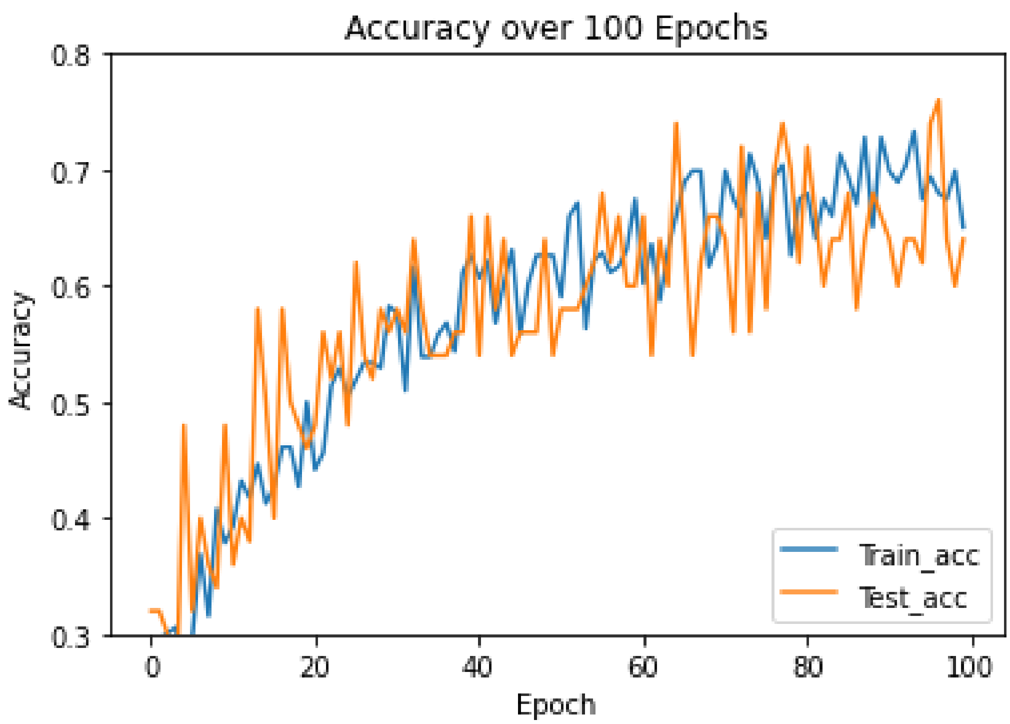
[5 points] Plot Accuracy and Loss During Training

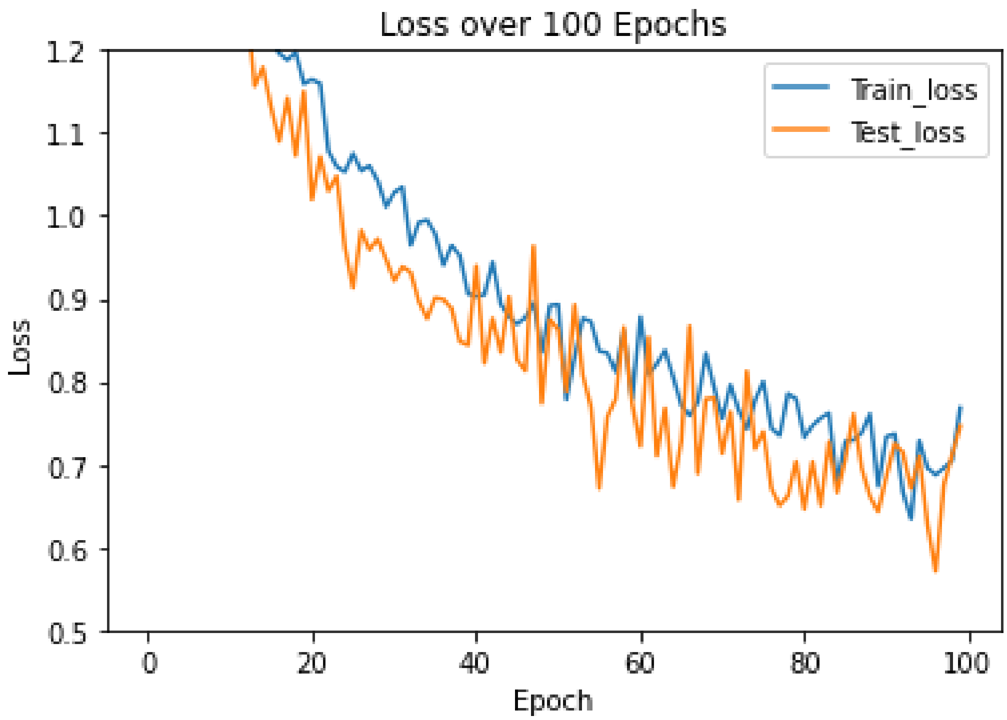
[18]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Train_acc')
plt.plot(history.history['val_accuracy'], label = 'Test_acc')
plt.xlabel('Epoch')
```

```python
plt.ylabel('Accuracy')
plt.ylim([0.3, 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([0.5, 1.2])
plt.legend(loc='upper right')
plt.title('Loss over 100 Epochs')
plt.show()
```





Testing Model

[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,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 [==============================] - 4s 110ms/step - loss: 0.8244 -
accuracy: 0.6111
Test loss: 0.8243839740753174
Test accuracy: 0.6111111044883728
```

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.

[29]:
```python
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 = intermediate_layer_model.predict(tsne_eval_generator)

intermediate_tsne = TSNE(n_components=2, n_iter=1000, verbose=1).
 ↪fit_transform(intermediate)

actual_classes = tsne_eval_generator.classes
actual_colors = []
for i in actual_classes:
    if i == 0:
        actual_colors.append('red')
    elif i == 1:
        actual_colors.append('blue')
    elif i == 2:
        actual_colors.append('yellow')
    else:
        actual_colors.append('green')

# in actual_classes list:
# 0 - 60 = covid - 0
# 61 - 130 = Normal - 1
# 131 - 200 = Pnemonia bacterial - 2
# 201 - 270 = Pnemonia Viral - 3

x = intermediate_tsne[:,0]
y = intermediate_tsne[:,1]

plt.figure(figsize=(8, 8))
plt.scatter(x, y, color = actual_colors)
plt.scatter(x[0], y[0], color = actual_colors[0], label = 'Covid-19')
plt.scatter(x[65], y[65], color = actual_colors[65], label = 'Normal')
```

```
plt.scatter(x[140], y[140], color = actual_colors[140], label = 'Pneumonia␣
 ↪Bacterial')
plt.scatter(x[210], y[210], color = actual_colors[210], label = 'Pneumonia␣
 ↪Viral')
plt.legend(loc='upper left')
plt.show()
```

```
Found 270 images belonging to 4 classes.
[t-SNE] Computing 91 nearest neighbors…
[t-SNE] Indexed 270 samples in 0.004s…
[t-SNE] Computed neighbors for 270 samples in 0.083s…
[t-SNE] Computed conditional probabilities for sample 270 / 270
[t-SNE] Mean sigma: 1.866734
[t-SNE] KL divergence after 250 iterations with early exaggeration: 62.831902
[t-SNE] KL divergence after 1000 iterations: 0.451694
```