Ziqi_Tan_task2

April 22, 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

from tensorflow.python.client import device_lib

print(device_lib.list_local_devices())

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

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 8148909400042469667
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 4930941747
locality {
  bus_id: 1
  links {
  }
}
incarnation: 17748192500196653471
physical_device_desc: "device: 0, name: GeForce GTX 1060, pci bus id:
0000:01:00.0, compute capability: 6.1"
]
```

[1]: '2.1.0'

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")
```

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

```
C:\Users\tanzi\Anaconda3\lib\site-
packages\keras_preprocessing\image\image_data_generator.py:341: UserWarning:
This ImageDataGenerator specifies `zca_whitening` which overrides setting
of`featurewise_std_normalization`.
  warnings.warn('This ImageDataGenerator specifies '
```

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

[4]:
```
# raise NotImplementedError("Build your model based on an architecture of your␣
 ↪choice "
#                           "A sample model summary is shown below")

# Implement VGG16
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Flatten, Dense, Dropout
```

```python
from tensorflow.keras.models import Sequential

vgg_16 = VGG16(include_top=False, weights='imagenet', input_shape=(224, 224, 3),␣
 ↪pooling='None', classes=4)
print(vgg_16.summary())
vgg_16.trainable = False

covid_model = Sequential()
covid_model.add(vgg_16)
covid_model.add(Flatten())
covid_model.add(Dropout(0.4))
covid_model.add(Dense(2048, activation='relu'))
covid_model.add(Dropout(0.3))
covid_model.add(Dense(256, activation='relu'))
covid_model.add(Dense(4, activation='softmax'))

covid_model.build(input_shape=(224, 224, 3))
covid_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
```

```
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
None
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dropout (Dropout)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 2048)              51382272
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               524544
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028
=================================================================
Total params: 66,622,532
Trainable params: 51,907,844
Non-trainable params: 14,714,688
_________________________________________________________________
```

[5 points] Train Model

[5]:
```
# FIT MODEL
from tensorflow.keras.optimizers import SGD
print(len(train_batches))
```

```python
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")
# Best Accuracy:
covid_model.compile(optimizer='adam', loss=tf.keras.losses.
 ↪CategoricalCrossentropy(from_logits=False), metrics=['accuracy'])

# print the device library
print(device_lib.list_local_devices())

history = None

with tf.device("GPU:0"):
    # history = covid_model.fit(train_batches, epochs=100,␣
 ↪validation_data=(valid_batches))
    history = covid_model.fit_generator(generator=train_batches,
                              steps_per_epoch=STEP_SIZE_TRAIN,
                              epochs=100,
                              validation_data=(valid_batches),
                              validation_steps=STEP_SIZE_VALID)
```

```
22
6
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 16475019960960570557
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 4930941747
locality {
  bus_id: 1
  links {
  }
}
incarnation: 12251531707959702585
physical_device_desc: "device: 0, name: GeForce GTX 1060, pci bus id:
0000:01:00.0, compute capability: 6.1"
]
WARNING:tensorflow:From <ipython-input-5-93010db56821>:24: Model.fit_generator
(from tensorflow.python.keras.engine.training) is deprecated and will be removed
in a future version.
```

```
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
```

```
C:\Users\tanzi\Anaconda3\lib\site-
packages\keras_preprocessing\image\image_data_generator.py:716: 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 '
C:\Users\tanzi\Anaconda3\lib\site-
packages\keras_preprocessing\image\image_data_generator.py:735: 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 '
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 21 steps, validate for 5 steps
Epoch 1/100
```

```
C:\Users\tanzi\Anaconda3\lib\site-
packages\keras_preprocessing\image\image_data_generator.py:716: 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 '
C:\Users\tanzi\Anaconda3\lib\site-
packages\keras_preprocessing\image\image_data_generator.py:735: 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 '
```

```
21/21 [==============================] - 8s 370ms/step - loss: 3.8587 -
accuracy: 0.3495 - val_loss: 1.1898 - val_accuracy: 0.4600
Epoch 2/100
21/21 [==============================] - 6s 286ms/step - loss: 2.0441 -
accuracy: 0.3981 - val_loss: 1.8222 - val_accuracy: 0.5600
Epoch 3/100
21/21 [==============================] - 6s 288ms/step - loss: 1.5520 -
accuracy: 0.4272 - val_loss: 1.1167 - val_accuracy: 0.4600
Epoch 4/100
21/21 [==============================] - 6s 291ms/step - loss: 1.4100 -
accuracy: 0.4175 - val_loss: 1.1131 - val_accuracy: 0.4600
Epoch 5/100
21/21 [==============================] - 6s 285ms/step - loss: 1.3205 -
accuracy: 0.4466 - val_loss: 1.2768 - val_accuracy: 0.4800
```

```
Epoch 6/100
21/21 [==============================] - 6s 289ms/step - loss: 1.2613 -
accuracy: 0.4466 - val_loss: 0.8208 - val_accuracy: 0.6000
Epoch 7/100
21/21 [==============================] - 6s 304ms/step - loss: 0.9607 -
accuracy: 0.5437 - val_loss: 0.8576 - val_accuracy: 0.5600
Epoch 8/100
21/21 [==============================] - 6s 306ms/step - loss: 1.0339 -
accuracy: 0.5340 - val_loss: 0.9405 - val_accuracy: 0.5800
Epoch 9/100
21/21 [==============================] - 7s 324ms/step - loss: 0.9154 -
accuracy: 0.6019 - val_loss: 0.8321 - val_accuracy: 0.5600
Epoch 10/100
21/21 [==============================] - 6s 293ms/step - loss: 0.9540 -
accuracy: 0.5874 - val_loss: 0.8509 - val_accuracy: 0.6200
Epoch 11/100
21/21 [==============================] - 6s 308ms/step - loss: 0.9162 -
accuracy: 0.5243 - val_loss: 0.7800 - val_accuracy: 0.6800
Epoch 12/100
21/21 [==============================] - 6s 307ms/step - loss: 0.9532 -
accuracy: 0.5667 - val_loss: 0.9397 - val_accuracy: 0.5200
Epoch 13/100
21/21 [==============================] - 6s 297ms/step - loss: 0.8424 -
accuracy: 0.6311 - val_loss: 0.7853 - val_accuracy: 0.6200
Epoch 14/100
21/21 [==============================] - 6s 295ms/step - loss: 0.9242 -
accuracy: 0.6165 - val_loss: 0.7193 - val_accuracy: 0.7200
Epoch 15/100
21/21 [==============================] - 6s 292ms/step - loss: 0.8644 -
accuracy: 0.5922 - val_loss: 0.8498 - val_accuracy: 0.5800
Epoch 16/100
21/21 [==============================] - 7s 323ms/step - loss: 0.9213 -
accuracy: 0.5825 - val_loss: 0.9498 - val_accuracy: 0.5200
Epoch 17/100
21/21 [==============================] - 6s 309ms/step - loss: 0.8941 -
accuracy: 0.6165 - val_loss: 0.6927 - val_accuracy: 0.7400
Epoch 18/100
21/21 [==============================] - 7s 332ms/step - loss: 0.9299 -
accuracy: 0.5728 - val_loss: 0.8056 - val_accuracy: 0.6200
Epoch 19/100
21/21 [==============================] - 7s 327ms/step - loss: 0.8936 -
accuracy: 0.6165 - val_loss: 0.7002 - val_accuracy: 0.6600
Epoch 20/100
21/21 [==============================] - 6s 302ms/step - loss: 0.8612 -
accuracy: 0.5874 - val_loss: 0.7184 - val_accuracy: 0.6400
Epoch 21/100
21/21 [==============================] - 6s 305ms/step - loss: 0.8224 -
accuracy: 0.6262 - val_loss: 0.6335 - val_accuracy: 0.7200
```

```
Epoch 22/100
21/21 [==============================] - 6s 302ms/step - loss: 0.8099 -
accuracy: 0.6408 - val_loss: 0.6922 - val_accuracy: 0.6800
Epoch 23/100
21/21 [==============================] - 7s 313ms/step - loss: 0.7974 -
accuracy: 0.6311 - val_loss: 0.6061 - val_accuracy: 0.6800
Epoch 24/100
21/21 [==============================] - 7s 328ms/step - loss: 0.7831 -
accuracy: 0.6456 - val_loss: 0.6088 - val_accuracy: 0.7400
Epoch 25/100
21/21 [==============================] - 7s 343ms/step - loss: 0.7973 -
accuracy: 0.6214 - val_loss: 0.7293 - val_accuracy: 0.7000
Epoch 26/100
21/21 [==============================] - 7s 310ms/step - loss: 0.7863 -
accuracy: 0.6262 - val_loss: 0.7739 - val_accuracy: 0.6600
Epoch 27/100
21/21 [==============================] - 6s 303ms/step - loss: 0.7823 -
accuracy: 0.6845 - val_loss: 0.8004 - val_accuracy: 0.5600
Epoch 28/100
21/21 [==============================] - 7s 320ms/step - loss: 0.7809 -
accuracy: 0.6505 - val_loss: 0.7135 - val_accuracy: 0.6000
Epoch 29/100
21/21 [==============================] - 7s 337ms/step - loss: 0.8037 -
accuracy: 0.6456 - val_loss: 0.7832 - val_accuracy: 0.6200
Epoch 30/100
21/21 [==============================] - 7s 333ms/step - loss: 0.7879 -
accuracy: 0.6748 - val_loss: 0.7098 - val_accuracy: 0.6800
Epoch 31/100
21/21 [==============================] - 7s 325ms/step - loss: 0.8194 -
accuracy: 0.6650 - val_loss: 0.6194 - val_accuracy: 0.7200
Epoch 32/100
21/21 [==============================] - 7s 313ms/step - loss: 0.7809 -
accuracy: 0.6456 - val_loss: 0.5439 - val_accuracy: 0.7200
Epoch 33/100
21/21 [==============================] - 7s 328ms/step - loss: 0.6872 -
accuracy: 0.7233 - val_loss: 0.7928 - val_accuracy: 0.6000
Epoch 34/100
21/21 [==============================] - 7s 329ms/step - loss: 0.7159 -
accuracy: 0.6845 - val_loss: 0.6797 - val_accuracy: 0.7000
Epoch 35/100
21/21 [==============================] - 7s 327ms/step - loss: 0.7249 -
accuracy: 0.6845 - val_loss: 0.7856 - val_accuracy: 0.5600
Epoch 36/100
21/21 [==============================] - 6s 299ms/step - loss: 0.8282 -
accuracy: 0.6505 - val_loss: 0.6439 - val_accuracy: 0.6600
Epoch 37/100
21/21 [==============================] - 6s 295ms/step - loss: 0.6671 -
accuracy: 0.6845 - val_loss: 0.5900 - val_accuracy: 0.7800
```

```
Epoch 38/100
21/21 [==============================] - 6s 301ms/step - loss: 0.6975 -
accuracy: 0.6602 - val_loss: 0.6348 - val_accuracy: 0.7400
Epoch 39/100
21/21 [==============================] - 6s 297ms/step - loss: 0.6470 -
accuracy: 0.7087 - val_loss: 0.6431 - val_accuracy: 0.6800
Epoch 40/100
21/21 [==============================] - 6s 297ms/step - loss: 0.7934 -
accuracy: 0.6311 - val_loss: 0.7585 - val_accuracy: 0.6800
Epoch 41/100
21/21 [==============================] - 6s 299ms/step - loss: 0.8159 -
accuracy: 0.6165 - val_loss: 0.6815 - val_accuracy: 0.7000
Epoch 42/100
21/21 [==============================] - 6s 303ms/step - loss: 0.8345 -
accuracy: 0.6117 - val_loss: 0.5631 - val_accuracy: 0.8000
Epoch 43/100
21/21 [==============================] - 6s 302ms/step - loss: 0.7545 -
accuracy: 0.6796 - val_loss: 0.5804 - val_accuracy: 0.7800
Epoch 44/100
21/21 [==============================] - 7s 331ms/step - loss: 0.7569 -
accuracy: 0.6456 - val_loss: 0.7406 - val_accuracy: 0.6000
Epoch 45/100
21/21 [==============================] - 7s 314ms/step - loss: 0.8799 -
accuracy: 0.5922 - val_loss: 0.9570 - val_accuracy: 0.5600
Epoch 46/100
21/21 [==============================] - 7s 330ms/step - loss: 0.7156 -
accuracy: 0.6602 - val_loss: 0.6582 - val_accuracy: 0.6000
Epoch 47/100
21/21 [==============================] - 7s 316ms/step - loss: 0.7574 -
accuracy: 0.6408 - val_loss: 0.6184 - val_accuracy: 0.7200
Epoch 48/100
21/21 [==============================] - 7s 332ms/step - loss: 0.7024 -
accuracy: 0.6942 - val_loss: 0.6061 - val_accuracy: 0.6800
Epoch 49/100
21/21 [==============================] - 7s 335ms/step - loss: 0.6875 -
accuracy: 0.7039 - val_loss: 0.6364 - val_accuracy: 0.6600
Epoch 50/100
21/21 [==============================] - 7s 313ms/step - loss: 0.6608 -
accuracy: 0.7233 - val_loss: 0.5682 - val_accuracy: 0.7200
Epoch 51/100
21/21 [==============================] - 7s 315ms/step - loss: 0.6426 -
accuracy: 0.6845 - val_loss: 0.6027 - val_accuracy: 0.7200
Epoch 52/100
21/21 [==============================] - 7s 316ms/step - loss: 0.6916 -
accuracy: 0.6748 - val_loss: 0.5864 - val_accuracy: 0.6800
Epoch 53/100
21/21 [==============================] - 7s 312ms/step - loss: 0.6819 -
accuracy: 0.7136 - val_loss: 0.5381 - val_accuracy: 0.6600
```

```
Epoch 54/100
21/21 [==============================] - 6s 304ms/step - loss: 0.7215 -
accuracy: 0.6893 - val_loss: 0.5720 - val_accuracy: 0.6600
Epoch 55/100
21/21 [==============================] - 6s 308ms/step - loss: 0.6529 -
accuracy: 0.7087 - val_loss: 0.7006 - val_accuracy: 0.7000
Epoch 56/100
21/21 [==============================] - 7s 336ms/step - loss: 0.6582 -
accuracy: 0.7379 - val_loss: 0.9457 - val_accuracy: 0.5600
Epoch 57/100
21/21 [==============================] - 7s 342ms/step - loss: 0.6781 -
accuracy: 0.6942 - val_loss: 0.7476 - val_accuracy: 0.6800
Epoch 58/100
21/21 [==============================] - 7s 321ms/step - loss: 0.8233 -
accuracy: 0.6602 - val_loss: 0.6619 - val_accuracy: 0.7000
Epoch 59/100
21/21 [==============================] - 7s 331ms/step - loss: 0.6612 -
accuracy: 0.7282 - val_loss: 0.5853 - val_accuracy: 0.6800
Epoch 60/100
21/21 [==============================] - 7s 329ms/step - loss: 0.6325 -
accuracy: 0.7524 - val_loss: 0.6752 - val_accuracy: 0.6800
Epoch 61/100
21/21 [==============================] - 7s 323ms/step - loss: 0.7368 -
accuracy: 0.6796 - val_loss: 0.7286 - val_accuracy: 0.6000
Epoch 62/100
21/21 [==============================] - 7s 328ms/step - loss: 0.7283 -
accuracy: 0.6553 - val_loss: 0.5970 - val_accuracy: 0.6800
Epoch 63/100
21/21 [==============================] - 8s 385ms/step - loss: 0.6396 -
accuracy: 0.7136 - val_loss: 0.5369 - val_accuracy: 0.6600
Epoch 64/100
21/21 [==============================] - 7s 330ms/step - loss: 0.6504 -
accuracy: 0.7039 - val_loss: 0.7590 - val_accuracy: 0.6400
Epoch 65/100
21/21 [==============================] - 7s 331ms/step - loss: 0.7254 -
accuracy: 0.6505 - val_loss: 0.7043 - val_accuracy: 0.6400
Epoch 66/100
21/21 [==============================] - 7s 337ms/step - loss: 0.6929 -
accuracy: 0.6857 - val_loss: 0.5924 - val_accuracy: 0.6400
Epoch 67/100
21/21 [==============================] - 7s 319ms/step - loss: 0.6208 -
accuracy: 0.7087 - val_loss: 0.5933 - val_accuracy: 0.7200
Epoch 68/100
21/21 [==============================] - 7s 319ms/step - loss: 0.6414 -
accuracy: 0.7233 - val_loss: 0.6660 - val_accuracy: 0.6400
Epoch 69/100
21/21 [==============================] - 7s 322ms/step - loss: 0.6678 -
accuracy: 0.6990 - val_loss: 0.8315 - val_accuracy: 0.6000
```

```
Epoch 70/100
21/21 [==============================] - 7s 316ms/step - loss: 0.7031 -
accuracy: 0.6505 - val_loss: 0.6619 - val_accuracy: 0.6800
Epoch 71/100
21/21 [==============================] - 7s 328ms/step - loss: 0.6461 -
accuracy: 0.6990 - val_loss: 0.5982 - val_accuracy: 0.6600
Epoch 72/100
21/21 [==============================] - 7s 326ms/step - loss: 0.6104 -
accuracy: 0.7233 - val_loss: 0.6970 - val_accuracy: 0.6200
Epoch 73/100
21/21 [==============================] - 6s 307ms/step - loss: 0.6344 -
accuracy: 0.7039 - val_loss: 0.5784 - val_accuracy: 0.6800
Epoch 74/100
21/21 [==============================] - 6s 300ms/step - loss: 0.7281 -
accuracy: 0.6553 - val_loss: 0.8764 - val_accuracy: 0.6000
Epoch 75/100
21/21 [==============================] - 7s 338ms/step - loss: 0.7901 -
accuracy: 0.6553 - val_loss: 0.9124 - val_accuracy: 0.6000
Epoch 76/100
21/21 [==============================] - 7s 316ms/step - loss: 0.7149 -
accuracy: 0.6796 - val_loss: 0.7966 - val_accuracy: 0.5800
Epoch 77/100
21/21 [==============================] - 6s 309ms/step - loss: 0.6509 -
accuracy: 0.6845 - val_loss: 0.5494 - val_accuracy: 0.7600
Epoch 78/100
21/21 [==============================] - 7s 317ms/step - loss: 0.6340 -
accuracy: 0.7184 - val_loss: 0.5362 - val_accuracy: 0.7400
Epoch 79/100
21/21 [==============================] - 7s 310ms/step - loss: 0.6719 -
accuracy: 0.6990 - val_loss: 0.7533 - val_accuracy: 0.6400
Epoch 80/100
21/21 [==============================] - 7s 314ms/step - loss: 0.6150 -
accuracy: 0.7087 - val_loss: 0.4776 - val_accuracy: 0.7600
Epoch 81/100
21/21 [==============================] - 6s 309ms/step - loss: 0.6313 -
accuracy: 0.7136 - val_loss: 0.5546 - val_accuracy: 0.7200
Epoch 82/100
21/21 [==============================] - 6s 308ms/step - loss: 0.6292 -
accuracy: 0.7136 - val_loss: 0.5645 - val_accuracy: 0.7000
Epoch 83/100
21/21 [==============================] - 7s 320ms/step - loss: 0.6085 -
accuracy: 0.7136 - val_loss: 0.7518 - val_accuracy: 0.6800
Epoch 84/100
21/21 [==============================] - 7s 310ms/step - loss: 0.7134 -
accuracy: 0.7087 - val_loss: 0.5164 - val_accuracy: 0.7400
Epoch 85/100
21/21 [==============================] - 7s 319ms/step - loss: 0.6645 -
accuracy: 0.7136 - val_loss: 0.6991 - val_accuracy: 0.6400
```

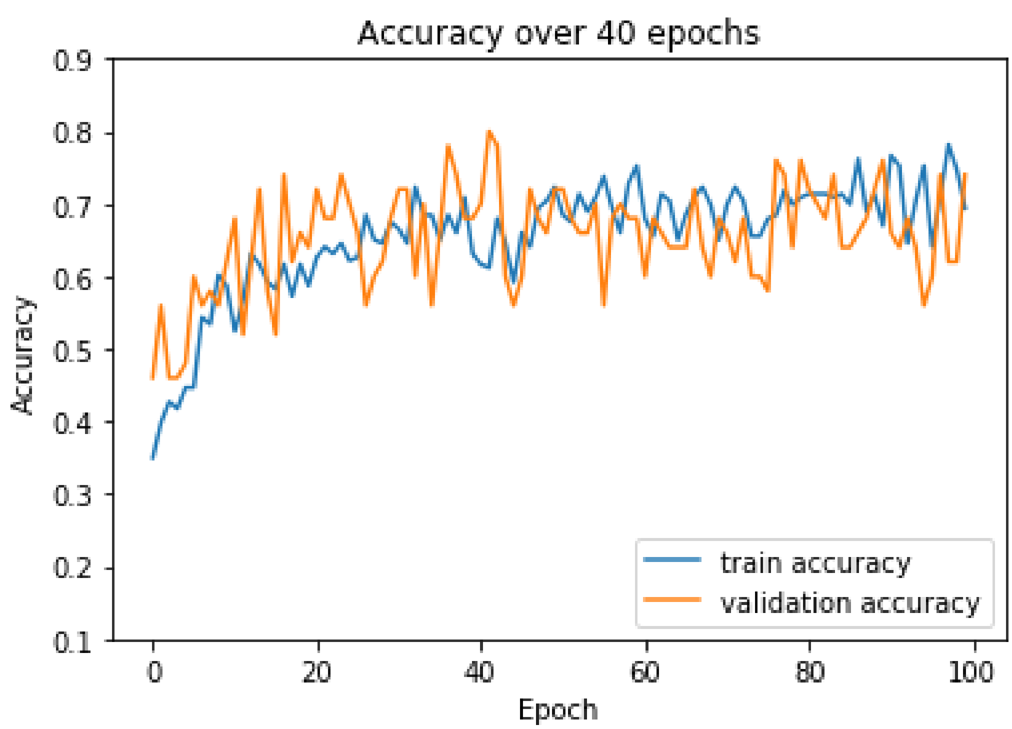
```
Epoch 86/100
21/21 [==============================] - 7s 333ms/step - loss: 0.6927 -
accuracy: 0.6990 - val_loss: 0.6768 - val_accuracy: 0.6400
Epoch 87/100
21/21 [==============================] - 7s 324ms/step - loss: 0.5986 -
accuracy: 0.7621 - val_loss: 0.6476 - val_accuracy: 0.6600
Epoch 88/100
21/21 [==============================] - 7s 322ms/step - loss: 0.6979 -
accuracy: 0.6893 - val_loss: 0.5447 - val_accuracy: 0.6800
Epoch 89/100
21/21 [==============================] - 7s 342ms/step - loss: 0.6768 -
accuracy: 0.7136 - val_loss: 0.5129 - val_accuracy: 0.7200
Epoch 90/100
21/21 [==============================] - 7s 328ms/step - loss: 0.7229 -
accuracy: 0.6699 - val_loss: 0.5712 - val_accuracy: 0.7600
Epoch 91/100
21/21 [==============================] - 7s 324ms/step - loss: 0.5355 -
accuracy: 0.7670 - val_loss: 0.5596 - val_accuracy: 0.6600
Epoch 92/100
21/21 [==============================] - 7s 315ms/step - loss: 0.6175 -
accuracy: 0.7524 - val_loss: 0.5598 - val_accuracy: 0.6400
Epoch 93/100
21/21 [==============================] - 7s 331ms/step - loss: 0.7596 -
accuracy: 0.6456 - val_loss: 0.5860 - val_accuracy: 0.6800
Epoch 94/100
21/21 [==============================] - 7s 340ms/step - loss: 0.6913 -
accuracy: 0.7039 - val_loss: 0.6611 - val_accuracy: 0.6400
Epoch 95/100
21/21 [==============================] - 7s 316ms/step - loss: 0.5960 -
accuracy: 0.7524 - val_loss: 0.7308 - val_accuracy: 0.5600
Epoch 96/100
21/21 [==============================] - 7s 315ms/step - loss: 0.7162 -
accuracy: 0.6408 - val_loss: 0.7640 - val_accuracy: 0.6000
Epoch 97/100
21/21 [==============================] - 7s 322ms/step - loss: 0.5884 -
accuracy: 0.7095 - val_loss: 0.6275 - val_accuracy: 0.7400
Epoch 98/100
21/21 [==============================] - 7s 321ms/step - loss: 0.5680 -
accuracy: 0.7816 - val_loss: 0.7004 - val_accuracy: 0.6200
Epoch 99/100
21/21 [==============================] - 7s 328ms/step - loss: 0.6383 -
accuracy: 0.7476 - val_loss: 0.8310 - val_accuracy: 0.6200
Epoch 100/100
21/21 [==============================] - 7s 338ms/step - loss: 0.7117 -
accuracy: 0.6942 - val_loss: 0.6388 - val_accuracy: 0.7400
```

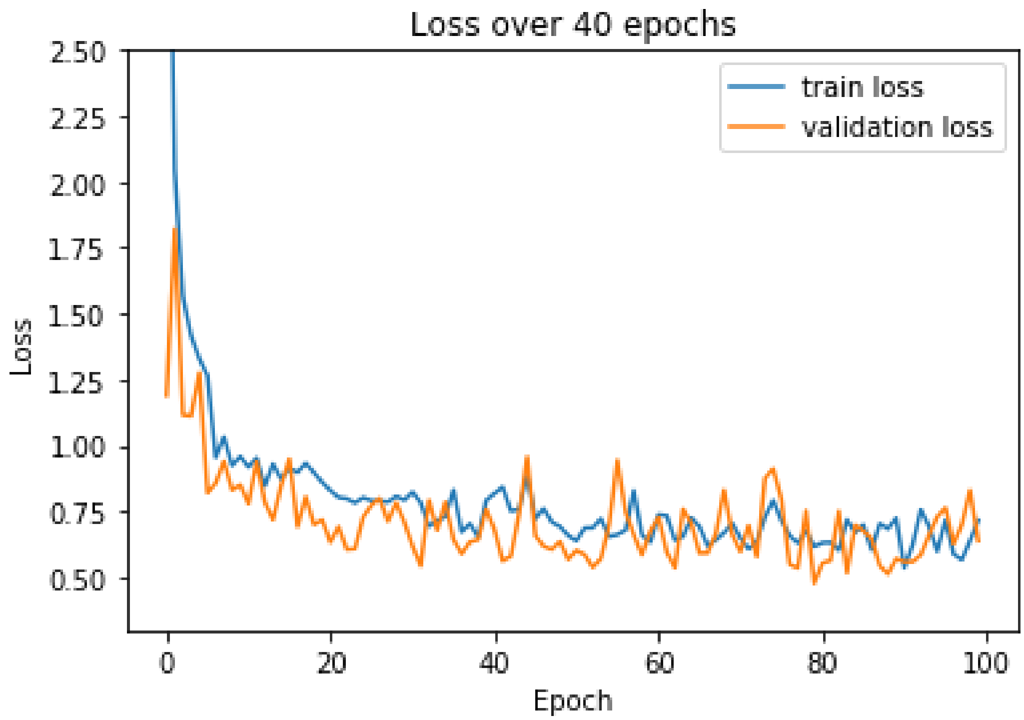
[5 points] Plot Accuracy and Loss During Training

[6]:

```
import matplotlib.pyplot as plt

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

# Accuracy over 40 Epochs
plt.figure()
plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label = 'validation accuracy')
plt.title('Accuracy over 40 epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.1, 0.9])
plt.legend(loc='lower right')

# Loss over 40 Epochs
plt.figure()
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label = 'validation loss')
plt.title('Loss over 40 epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.ylim([0.3, 2.5])
plt.legend(loc='upper right')
```

[6]: <matplotlib.legend.Legend at 0x260a0666358>





Testing Model

[7]:
```
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.
 ↪flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                              ␣
 ↪batch_size=1,shuffle=True,seed=42,class_mode="categorical")
eval_generator.reset()
print(len(eval_generator))
x = covid_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
WARNING:tensorflow:From <ipython-input-7-dedefa902e64>:8:
Model.evaluate_generator (from tensorflow.python.keras.engine.training) is
deprecated and will be removed in a future version.
```

```
Instructions for updating:
Please use Model.evaluate, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
36/36 [==============================] - 1s 32ms/step - loss: 0.7699 - accuracy:
0.7500
Test loss: 0.7698995597610419
Test accuracy: 0.75
```

2.4 [10 points] TSNE Plot

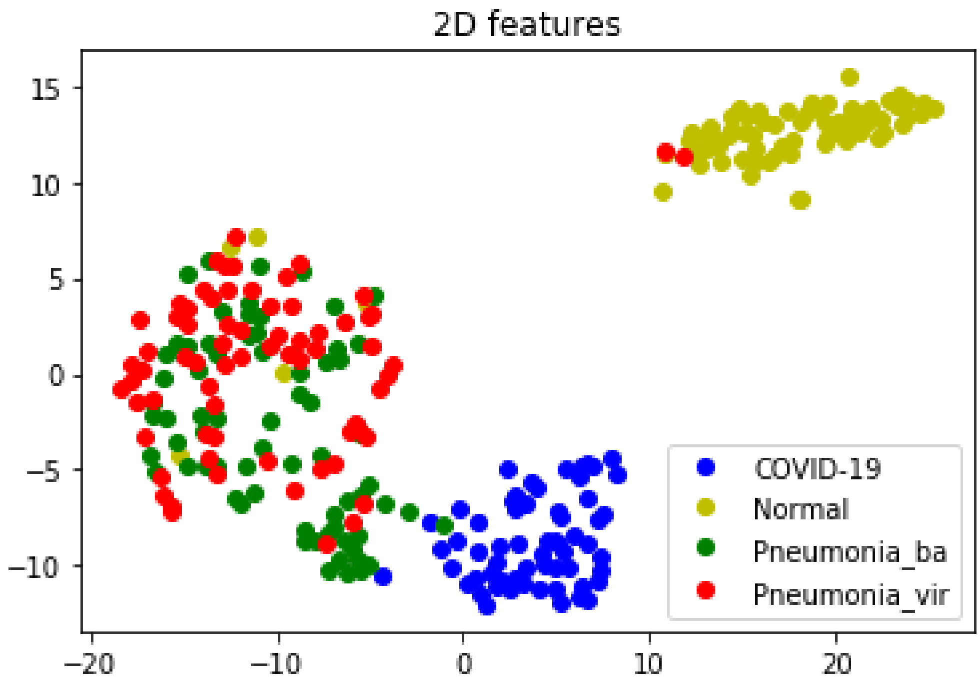
t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely used technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. After training is complete, extract features from a specific deep layer of your choice, use t-SNE to reduce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

[9]:
```
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.models.Model(inputs=covid_model.input,
                                        outputs=covid_model.get_layer('dense_1').
 ↪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")

# raise NotImplementedError("Extract features from the tsne_data_generator and␣
 ↪fit a t-SNE model for the features,"
#                           "and plot the resulting 2D features of the four␣
 ↪classes.")

outputs = intermediate_layer_model.
 ↪predict_generator(tsne_eval_generator,270,verbose=1)
print(outputs.shape)
label = tsne_eval_generator.classes
features = TSNE(n_components=2).fit_transform(outputs)
print(features.shape)

covid_x = []
covid_y = []
normal_x = []
normal_y = []
```

```python
pneumonia_bac_x = []
pneumonia_bac_y = []
pneumonia_vir_x = []
pneumonia_vir_y = []

plt.figure()
for index in range(len(features)):
    if label[index] == 0:
        # COVID: Blue
        covid_x.append(features[index, 0])
        covid_y.append(features[index, 1])
    elif label[index] == 1:
        # Normal: Yellow
        normal_x.append(features[index, 0])
        normal_y.append(features[index, 1])
    elif label[index] == 2:
        # Pneumonia_bac: Green
        pneumonia_bac_x.append(features[index, 0])
        pneumonia_bac_y.append(features[index, 1])
    else:
        # Pneumonia_vir: Red
        pneumonia_vir_x.append(features[index, 0])
        pneumonia_vir_y.append(features[index, 1])

plt.title('2D features')
plt.plot(covid_x, covid_y, 'bo', label="COVID-19")
plt.plot(normal_x, normal_y, 'yo', label="Normal")
plt.plot(pneumonia_bac_x, pneumonia_bac_y, 'go', label="Pneumonia_ba")
plt.plot(pneumonia_vir_x, pneumonia_vir_y, 'ro', label="Pneumonia_vir")
plt.legend(loc='lower right')
```

```
Found 270 images belonging to 4 classes.
270/270 [==============================] - 6s 21ms/step
(270, 256)
(270, 2)
```

```
[9]: <matplotlib.legend.Legend at 0x261ed2848d0>
```


2D features
15
10
5
0
−5
−10
−20
−10
0
10
20
COVID-19
Normal
Pneumonia_ba
Pneumonia_vir

[ ]: