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.

```
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__

    '2.8.0'
```

Load Image Data

```
from google.colab import drive drive.mount('/content/drive')
```

```
DATA_LIST = os.listdir('/content/drive/My Drive/CC/Covid_Data_GradientCrescent/'
DATASET_PATH  = '/content/drive/My Drive/CC/Covid_Data_GradientCrescent/all/trai
TEST_DIR =  '/content/drive/My Drive/CC/Covid_Data_GradientCrescent/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
NUM_EPOCHS    = 100
LEARNING_RATE = 0.0001 # start off with high rate first 0.001 and experiment wit
```

```
     Drive already mounted at /content/drive; to attempt to forcibly remount, ca
```

Generate Training and Validation Batches

```
train_datagen = ImageDataGenerator(rescale=1./255,rotation_range=50,featurewise_
featurewise_std_normalization = True,width_sh
height_shift_range=0.2,shear_range=0.25,zoom_
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
shuffle=True,batch_size=BATCH_
subset = "training",seed=42,
class_mode="categorical")

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

```
     Found 216 images belonging to 4 classes.
     Found 54 images belonging to 4 classes.
       /usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data
       warnings.warn('This ImageDataGenerator specifies '
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.layers import Dropout from
tensorflow.keras.layers import Flatten from
tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dense from
tensorflow.keras.layers import Input from
tensorflow.keras.layers import AveragePooling2D from
tensorflow.keras.optimizers import Adam
import numpy as np
import argparse
```

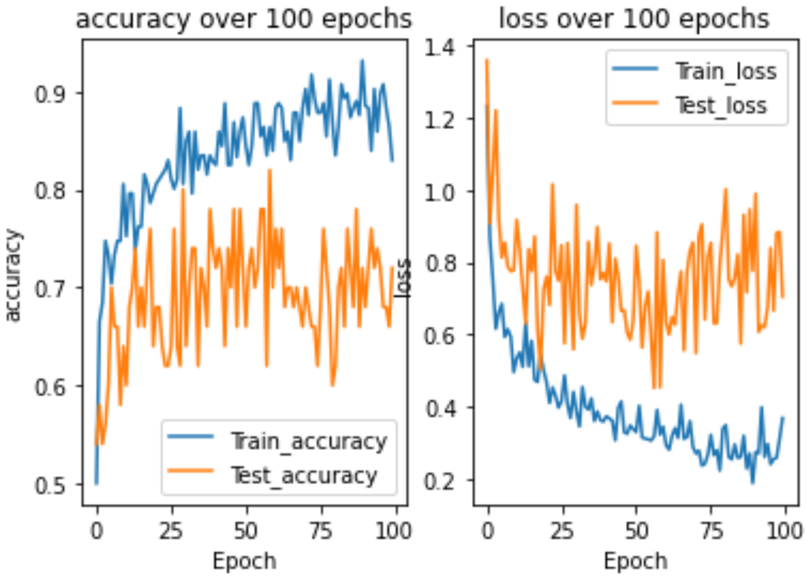
```
# model 2
model = tf.keras.models.Sequential()
model.add (DenseNet121 (weights= 'imagenet', include_top=False, input_shape =(22
model.add (BatchNormalization())
model.add (AveragePooling2D(pool_size=(2,2))) model.add
(Flatten())
#model.add (Dropout(0.3))
model.add(Dense(units=128,activation="relu"))
#model.add (Dropout(0.3))
model.add(Dense(units=4,activation="softmax"))
model.layers[0].trainable = False model.summary()
```

```
     Model: "sequential_3"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
     densenet121 (Functional)    (None, 7, 7, 1024)        7037504
     batch_normalization_3 (Batc  (None, 7, 7, 1024)       4096
     hNormalization)
     average_pooling2d_3 (Averag  (None, 3, 3, 1024)       0
     ePooling2D)
     flatten_3 (Flatten)         (None, 9216)              0
     dense_6 (Dense)             (None, 128)               1179776
     dense_7 (Dense)             (None, 4)                 516
     =================================================================
     Total params: 8,221,892
     Trainable params: 1,182,340
     Non-trainable params: 7,039,552
     _________________________________________________________________
```

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

STEP_SIZE_TRAIN=train_batches.n//train_batches.bat#    ch_size
STEP_SIZE_VALID=valid_batches.n//valid_batches.batch_size

model.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accura
history = model.fit(x=train_batches,epochs=NUM_EPOCHS,batch_size=BATCH_SIZE, ste
          validation_batch_size=BATCH_SIZE, validation_steps=STEP_SIZE_VALID)
validation_batch_size=BATCH_SIZE, validation_steps=STEP_SIZE_VALID)
     21/21 [==============================] - 41s 2s/step - loss: 0.3121 - accur
     Epoch 68/100
     21/21 [==============================] - 42s 2s/step - loss: 0.3175 - accur
     Epoch 69/100
     21/21 [==============================] - 43s 2s/step - loss: 0.3610 - accur
     Epoch 70/100
     21/21 [==============================] - 41s 2s/step - loss: 0.2897 - accur
     Epoch 71/100
     21/21 [==============================] - 41s 2s/step - loss: 0.2717 - accur
     Epoch 72/100
     21/21 [==============================] - 41s 2s/step - loss: 0.2798 - accur
     Epoch 73/100
```

```
21/21 [==============================] - 41s 2s/step - loss: 0.2384 - accur
Epoch 74/100
21/21 [==============================] - 41s 2s/step - loss: 0.2431 - accur
Epoch 75/100
21/21 [==============================] - 41s 2s/step - loss: 0.2684 - accur
Epoch 76/100
21/21 [==============================] - 41s 2s/step - loss: 0.3231 - accur
Epoch 77/100
21/21 [==============================] - 41s 2s/step - loss: 0.2673 - accur
Epoch 78/100
21/21 [==============================] - 41s 2s/step - loss: 0.2807 - accur
Epoch 79/100
21/21 [==============================] - 41s 2s/step - loss: 0.2246 - accur
Epoch 80/100
21/21 [==============================] - 41s 2s/step - loss: 0.3408 - accur
Epoch 81/100
21/21 [==============================] - 41s 2s/step - loss: 0.3500 - accur
Epoch 82/100
21/21 [==============================] - 41s 2s/step - loss: 0.2623 - accur
Epoch 83/100
21/21 [==============================] - 41s 2s/step - loss: 0.2553 - accur
Epoch 84/100
21/21 [==============================] - 41s 2s/step - loss: 0.2968 - accur
Epoch 85/100
21/21 [==============================] - 41s 2s/step - loss: 0.2595 - accur
Epoch 86/100
21/21 [==============================] - 41s 2s/step - loss: 0.2625 - accur
Epoch 87/100
21/21 [==============================] - 41s 2s/step - loss: 0.3205 - accur
Epoch 88/100
21/21 [==============================] - 41s 2s/step - loss: 0.2305 - accur
Epoch 89/100
21/21 [==============================] - 42s 2s/step - loss: 0.2716 - accur
Epoch 90/100
21/21 [==============================] - 43s 2s/step - loss: 0.1897 - accur
Epoch 91/100
21/21 [==============================] - 44s 2s/step - loss: 0.2763 - accur
Epoch 92/100
21/21 [==============================] - 43s 2s/step - loss: 0.2719 - accur
Epoch 93/100
21/21 [==============================] - 43s 2s/step - loss: 0.3995 - accur
Epoch 94/100
Epoch 94/100
21/21 [==============================] - 43s 2s/step - loss: 0.2620 - accur
Epoch 95/100
21/21 [==============================] - 43s 2s/step - loss: 0.2982 - accur
Epoch 96/100
21/21 [==============================] - 43s 2s/step - loss: 0.2423 - accur
Epoch 97/100
21/21 [==============================] - 43s 2s/step - loss: 0.2562 - accur
```

```
import·matplotlib.pyplot·as·plt

fig,·(ax)·=·plt.subplots(1,·2)

c=0

for·i·in·['accuracy',·'loss']:
··ax[c].plot(history.history[i],·label='Train_'+i)
··ax[c].plot(history.history['val_'+i],·label='Test_'+i)
··ax[c].set_xlabel('Epoch')  ··ax[c].set_ylabel(i)
··if·i=='accuracy':  ····ax[c].legend(loc='lower·right')
··else:
····ax[c].legend(loc='upper·right')

··ax[c].set_title(str(i)+'·over·'+str(NUM_EPOCHS)+'·epochs')
··c+=1

plt.show()
i
```



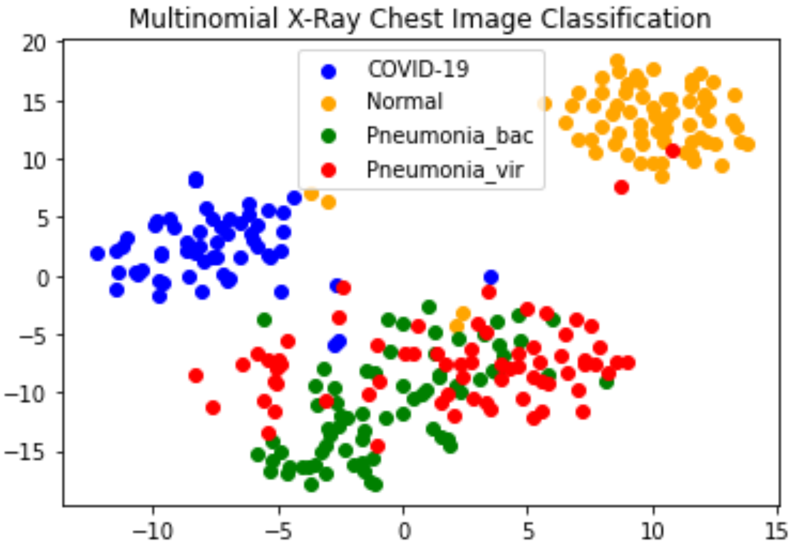
Testing Model

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

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZ
batch_size=1,shuffle=True,seed 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
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: UserWarning

36/36 [==============================] - 13s 361ms/step - loss: 0.4509 - ac
Test loss: 0.4508759081363678
Test accuracy: 0.8333333134651184
```

```
from sklearn.manifold import TSNE
from tensorflow.keras import models

intermediate_layer_model = models.Model(inputs=model.input,
                                        outputs=model.get_layer('dense_6').outpu

tsne_eval_generator = test_datagen.flow_from_directory(DATASET_PATH,target_size=
batch_size=1,shuffle=False,see

activations2 = intermediate_layer_model.predict(tsne_eval_generator, 1, verbose=
tsne2 = TSNE(n_components=2,init='random',random_state= 55)
print(activations2.shape)
tsne_obj2 = tsne2.fit_transform(activations2)t print(tsne_obj2.shape)
colors = ['blue', 'orange', 'green', 'red']
colors = ['blue', 'orange', 'green', 'red'] c=[colors[i]
for i in tsne_eval_generator.labels]
labels = ['COVID-19','Normal', 'Pneumonia_bac', 'Pneumonia_vir'] l=[labels[i]
for i in tsne_eval_generator.labels]

x1 = []
y1 = []

for i in range(len(labels)):
  x2 = []
  y2 = []
  for j in range(tsne_obj2.shape[0]):
    if(tsne_eval_generator.labels[j]==i):
      x2.append(tsne_obj2[j,0])
      y2.append(tsne_obj2[j,1])
  x1.append(x2)
y1.append(y2)

for i in range(len(labels)):
  plt.scatter(x1[i][:], y1[i],c = colors[i],label = labels[i])

plt.title("Multinomial X-Ray Chest Image Classification") plt.legend()
```

```
Found 270 images belonging to 4 classes.
270/270 [==============================] - 53s 186ms/step
(270, 128)
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793: Futu
  FutureWarning,
(270, 2)
<matplotlib.legend.Legend at 0x7f7b21b0af90>
```



1m 26s completed at 1:33 PM