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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", fo
rce_remount=True).
```

In [2]:
```
import tensorflow as tf
%tensorflow_version 2.x
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

In [3]:
```
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__
```

Out[3]:
```
'2.4.1'
```

### Load Image Data

In [4]:
```
DATA_LIST = os.listdir('/content/drive/MyDrive/cs542/all')
DATASET_PATH  = '/content/drive/MyDrive/cs542/all/train'
TEST_DIR =  '/content/drive/MyDrive/cs542/all/test'
IMAGE_SIZE    = (224, 224)
NUM_CLASSES   = len(DATA_LIST)
BATCH_SIZE    = 20  # try reducing batch size or freeze more layers if your GPU runs out of memory
NUM_EPOCHS    = 350
LEARNING_RATE = 0.00001 # start off with high rate first 0.001 and experiment with reducing it gradually
NUM_FREEZE1 = 150
NUM_FREEZE2 = 70
```

**Generate Training and Validation Batches**

In [5]:
```
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")
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data_generator.py:342: UserWarning: Thi
s 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.

In [ ]:
```
pretrained = tf.keras.applications.InceptionV3(
    include_top=False,
    weights="imagenet",
    input_shape=(224, 224, 3),
    classes=4
)

preprocess_input = tf.keras.applications.inception_v3.preprocess_input

print(f"Number of layers in the pretrained model: {len(pretrained.layers)}")

for i in range(NUM_FREEZE1):
    pretrained.layers[i].trainable = False
```

```
Number of layers in the pretrained model: 311
```

In [ ]:
```
training_layers = tf.keras.Sequential([
    # add more layers
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation = 'relu')
])

prediction_layer = tf.keras.layers.Dense(4, activation="softmax")
```

In [ ]:
```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = preprocess_input(inputs)
x = pretrained(x)
x = training_layers(x)
dense = tf.keras.layers.Dense(16, activation = 'relu')
x = dense(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
tf.math.truediv (TFOpLambda) (None, 224, 224, 3)       0
_________________________________________________________________
tf.math.subtract (TFOpLambda (None, 224, 224, 3)       0
_________________________________________________________________
inception_v3 (Functional)    (None, 5, 5, 2048)        21802784
_________________________________________________________________
sequential (Sequential)      (None, 128)               26280576
_________________________________________________________________
dense_3 (Dense)              (None, 16)                2064
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 68
=================================================================
Total params: 48,085,492
Trainable params: 43,649,684
Non-trainable params: 4,435,808
_________________________________________________________________
```

In [ ]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(lr=LEARNING_RATE),
              loss=tf.keras.losses.CategoricalCrossentropy(),
              metrics=['accuracy'])
```

**[5 points] Train Model**

In [ ]:
```
checkpoint_filepath1 = '/content/tmp/checkpoint1_1/'
checkpoint_filepath2 = '/content/tmp/checkpoint1_2/'
model_checkpoint_callback1 = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath1,
    sav_freq = 'epoch',
    save_weights_only=True,
    monitor='loss',
    mode='min',
    save_best_only=True)


model_checkpoint_callback2 = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath2,
    sav_freq = 'epoch',
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

In [ ]:
```
#Fit InceptionV3 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(train_batches,
                    epochs=NUM_EPOCHS,
                    validation_data=valid_batches,
                    callbacks = [model_checkpoint_callback1, model_checkpoint_callback2])
```

```
- val_accuracy: 0.5926
Epoch 328/350
11/11 [==============================] - 7s 619ms/step - loss: 0.7895 - accuracy: 0.6418 - val_loss: 0.7174
- val_accuracy: 0.7222
Epoch 329/350
11/11 [==============================] - 7s 628ms/step - loss: 0.8923 - accuracy: 0.6178 - val_loss: 0.7418
- val_accuracy: 0.6296
Epoch 330/350
11/11 [==============================] - 7s 649ms/step - loss: 0.7185 - accuracy: 0.6932 - val_loss: 0.7274
- val_accuracy: 0.6852
Epoch 331/350
11/11 [==============================] - 7s 636ms/step - loss: 0.6632 - accuracy: 0.6673 - val_loss: 0.6593
- val_accuracy: 0.5926
Epoch 332/350
11/11 [==============================] - 7s 644ms/step - loss: 0.8150 - accuracy: 0.5729 - val_loss: 0.7604
- val_accuracy: 0.6667
Epoch 333/350
11/11 [==============================] - 7s 638ms/step - loss: 0.6236 - accuracy: 0.7459 - val_loss: 0.8362
- val_accuracy: 0.6481
Epoch 334/350
```

In [ ]:
```
model.load_weights('/content/tmp/checkpoint1_1/')
```

Out[21]:
```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f96c620ac10>
```

In [21]:
```
checkpoint_filepath1 = '/content/tmp/checkpoint2_1/'
checkpoint_filepath2 = '/content/tmp/checkpoint2_2/'
checkpoint_filepath3 = '/content/tmp/checkpoint2_3/'
model_checkpoint_callback1 = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath1,
    sav_freq = 'epoch',
    save_weights_only=True,
    monitor='loss',
    mode='min',
    save_best_only=True)


model_checkpoint_callback2 = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath2,
    sav_freq = 'epoch',
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)

model_checkpoint_callback3 = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath3,
    sav_freq = 'epoch',
    save_weights_only=True,
    monitor='val_loss',
    mode='min',
    save_best_only=True)


pretrained2 = tf.keras.applications.MobileNet(
    include_top=False,
    weights="imagenet",
    input_shape=(224, 224, 3),
    classes=4
)

preprocess_input2 = tf.keras.applications.mobilenet.preprocess_input

print(f"Number of layers in the pretrained model: {len(pretrained2.layers)}")

for i in range(20):
    pretrained2.layers[i].trainable = False
```

```
training_layers2 = tf.keras.Sequential([
    # add more layers
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation = 'relu')
])

prediction_layer2 = tf.keras.layers.Dense(4, activation="softmax")

inputs2 = tf.keras.Input(shape=(None, None, 3))
x2 = preprocess_input2(inputs2)
x2 = pretrained2(x2)
x2 = training_layers2(x2)
dense2 = tf.keras.layers.Dense(32, activation = 'relu')
x2 = dense2(x2)
outputs2 = prediction_layer2(x2)
model2 = tf.keras.Model(inputs2, outputs2)
model2.summary()
```

```
Number of layers in the pretrained model: 86
Model: "model_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_14 (InputLayer)        [(None, None, None, 3)]   0
_________________________________________________________________
tf.math.truediv_6 (TFOpLambd (None, None, None, 3)     0
_________________________________________________________________
tf.math.subtract_6 (TFOpLamb (None, None, None, 3)     0
_________________________________________________________________
mobilenet_1.00_224 (Function (None, 7, 7, 1024)        3228864
_________________________________________________________________
sequential_6 (Sequential)    (None, 128)               25756288
_________________________________________________________________
dense_27 (Dense)             (None, 32)                4128
_________________________________________________________________
dense_26 (Dense)             (None, 4)                 132
=================================================================
Total params: 28,989,412
Trainable params: 28,953,508
```

```
Non-trainable params: 35,904
_________________________________________________________________
```

In [22]:
```
model2.compile(optimizer=tf.keras.optimizers.Adam(lr=0.00001),
               loss=tf.keras.losses.CategoricalCrossentropy(),
               metrics=['accuracy'])
```

In [23]:
```
#Fit MobileNet 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

history2 = model2.fit(train_batches,
                      epochs=300,
                      validation_data=valid_batches,
                      callbacks = [model_checkpoint_callback1,model_checkpoint_callback2,model_checkpoint_callback
```

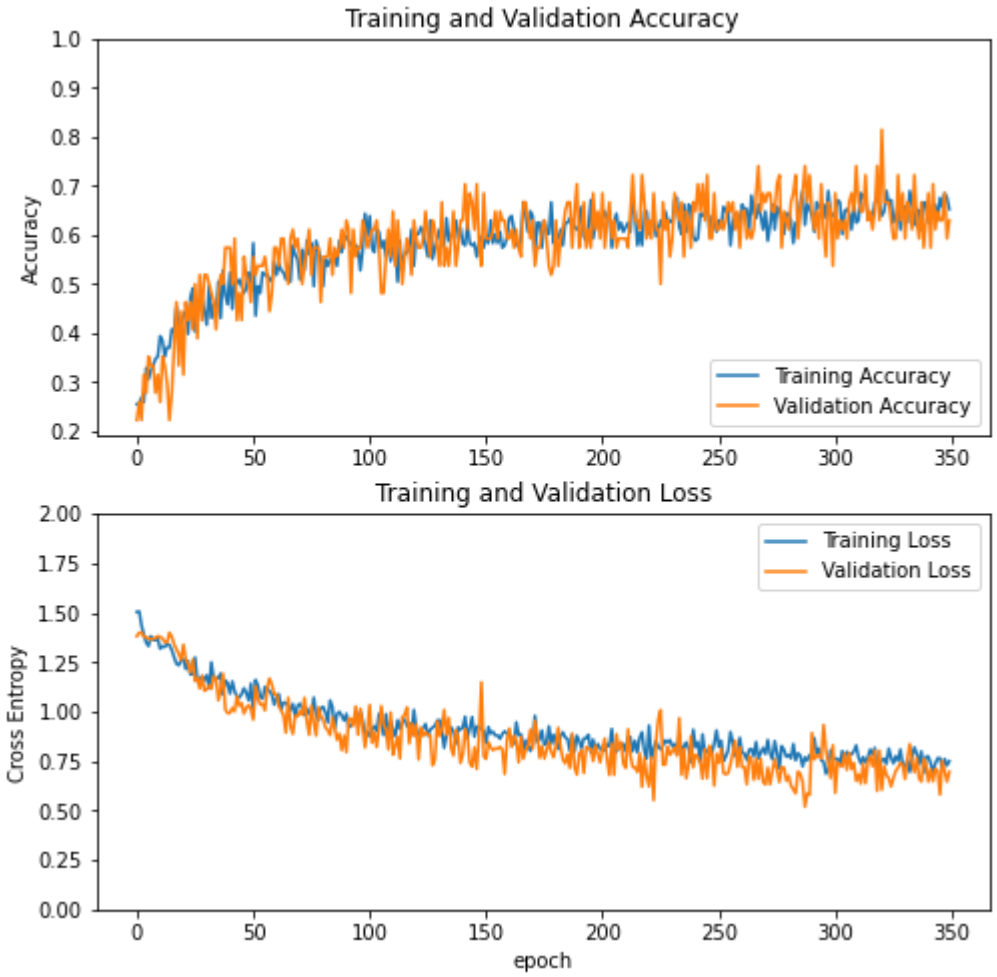
```
- val_accuracy: 0.6481
Epoch 282/300
11/11 [==============================] - 7s 668ms/step - loss: 0.5560 - accuracy: 0.7449 - val_loss: 0.7699
- val_accuracy: 0.5926
Epoch 283/300
11/11 [==============================] - 7s 650ms/step - loss: 0.5347 - accuracy: 0.7468 - val_loss: 0.9984
- val_accuracy: 0.5370
Epoch 284/300
11/11 [==============================] - 7s 650ms/step - loss: 0.5672 - accuracy: 0.7230 - val_loss: 1.1157
- val_accuracy: 0.6296
Epoch 285/300
11/11 [==============================] - 7s 673ms/step - loss: 0.4884 - accuracy: 0.7476 - val_loss: 1.1445
- val_accuracy: 0.5370
Epoch 286/300
11/11 [==============================] - 7s 673ms/step - loss: 0.5651 - accuracy: 0.7622 - val_loss: 0.7275
- val_accuracy: 0.6296
Epoch 287/300
11/11 [==============================] - 7s 669ms/step - loss: 0.4899 - accuracy: 0.7987 - val_loss: 0.8388
- val_accuracy: 0.5926
```

In [30]:
```
model2.load_weights('/content/tmp/checkpoint2_3/')
```

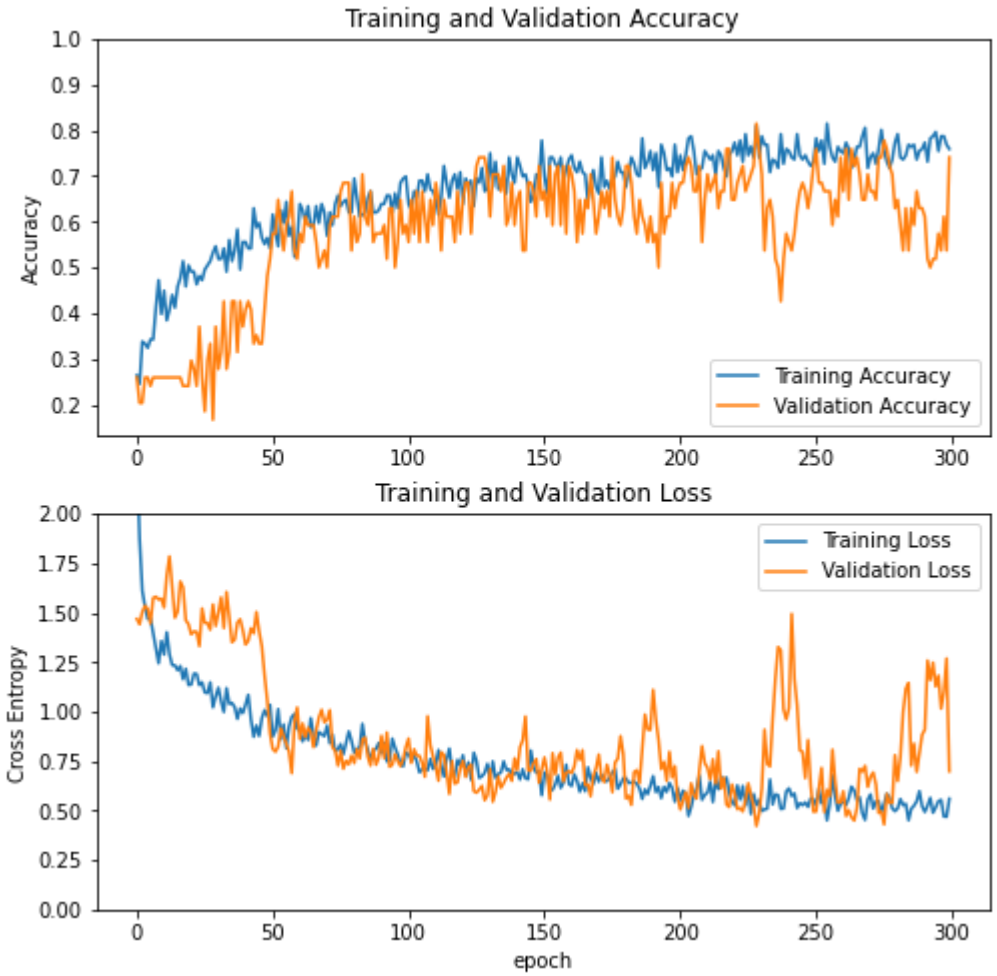
Out[30]:
```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f6e2918d950>
```

### [5 points] Plot Accuracy and Loss During Training

In [ ]:
```
#Plots of InceptionV3
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,2])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```


Training and Validation Accuracy
Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Cross Entropy
Training Loss
Validation Loss
epoch

In [24]:
```
#Plots of MobileNet
acc = history2.history['accuracy']
val_acc = history2.history['val_accuracy']

loss = history2.history['loss']
val_loss = history2.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,2.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```




**Testing Model**

In [ ]:
```
#Test accuracy for InceptionV3
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                  batch_size=1,shuffle=False,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
 2/36 [>.............................] - ETA: 2s - loss: 0.1211 - accuracy: 1.0000

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.ev
aluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which su
pports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '

36/36 [==============================] - 1s 26ms/step - loss: 0.8881 - accuracy: 0.7222
Test loss: 0.8880665302276611
Test accuracy: 0.7222222089767456
```

In [31]:
```
#Test accuracy for MobileNet
test_datagen = ImageDataGenerator(rescale=1. / 255)

eval_generator = test_datagen.flow_from_directory(TEST_DIR,target_size=IMAGE_SIZE,
                                                  batch_size=1,shuffle=False,seed=42,class_mode="categorical")
eval_generator.reset()
print(len(eval_generator))
x = model2.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
 2/36 [>.............................] - ETA: 2s - loss: 0.0430 - accuracy: 1.0000

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.ev
aluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which su
pports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '

36/36 [==============================] - 1s 22ms/step - loss: 0.8229 - accuracy: 0.5833
Test loss: 0.8229268193244934
Test accuracy: 0.5833333134651184
```

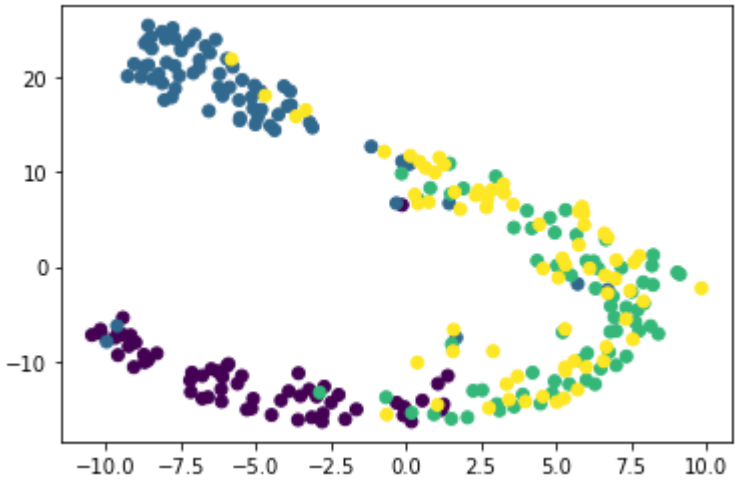
# [10 points] TSNE Plot

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 [ ]:
```
#TSNE Plot for InceptionV3
from sklearn.manifold import TSNE

intermediate_layer_model = tf.keras.models.Model(inputs=model.input,
                                         outputs=model.layers[-2].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_output = intermediate_layer_model.predict(tsne_eval_generator)

tsne = TSNE(n_components=2)
y = tsne.fit_transform(intermediate_output)

plt.scatter(y[:,0],y[:,1], c = tsne_eval_generator.labels)
```

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

Out[23]:
```
<matplotlib.collections.PathCollection at 0x7f960eb8c610>
```



In [36]:
```
#TSNE Plot for MobileNet
from sklearn.manifold import TSNE
intermediate_layer_model2 = tf.keras.models.Model(inputs=model2.input,
                                        outputs=model2.layers[-2].output)

tsne_eval_generator2 = test_datagen.flow_from_directory(DATASET_PATH,target_size=IMAGE_SIZE,
                                                        batch_size=1,shuffle=False,seed=42,class_mode="categorical")
intermediate_output2 = intermediate_layer_model2.predict(tsne_eval_generator2)

tsne2 = TSNE(n_components=2)
y2 = tsne2.fit_transform(intermediate_output2)

scatter = plt.scatter(y2[:,0],y2[:,1], c = tsne_eval_generator2.labels)
classes = ['covid','normal','pneumonia_bac','pneumonia_vir']
plt.legend(handles=scatter.legend_elements()[0], labels=classes)
```

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

Out[36]:
```
<matplotlib.legend.Legend at 0x7f6d55578190>
```