In [1]:

```
import keras
import numpy as np
from keras.applications import vgg16
from keras.models import Sequential, Model
from keras.layers import Dense, Flatten , Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.optimizers import Adam
import tensorflow as tf
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
            tf.config.experimental.set_virtual_device_configuration(
                        gpus[0],
            [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=2048)])
            logical_gpus = tf.config.experimental.list_logical_devices('GPU')
            print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:

      print(e)
```

```
Using TensorFlow backend.
```

```
1 Physical GPUs, 1 Logical GPUs
```

In [2]:

```
train_data_dir = "D:/Dataset/Train"
validation_data_dir = "D:/Dataset/Test"
batch_size = 13
img_height = 208
img_width = 176
numClasses=4
epoch=13
train_datagen= ImageDataGenerator(
        rescale= 1./255,
        shear_range= 0.2,
        zoom_range= 0.2,
        horizontal_flip= True)


validation_datagen= ImageDataGenerator(
        rescale= 1./255)
train_generator= train_datagen.flow_from_directory(
        train_data_dir, target_size= (img_height, img_width),
        batch_size= batch_size,
        class_mode='categorical')
validation_generator= validation_datagen.flow_from_directory(
        validation_data_dir, target_size= (img_height, img_width),
        batch_size= batch_size,
        class_mode= 'categorical')

base_model= keras.applications.vgg16.VGG16(weights= 'imagenet', include_top= False, input_shape=(im
g_height, img_width, 3
                                                                                                  ))
```

```
Found 5472 images belonging to 4 classes.
Found 1279 images belonging to 4 classes.
```

In [3]:

```
add_model= Sequential()
add_model.add(Flatten(input_shape=base_model.output_shape[1:]))
add_model.add(Dense(250, activation= 'relu'))
add_model.add(Dropout(0.2))
add_model.add(Dense(4, activation='softmax'))
model= Model(inputs= base_model.input, outputs=add_model(base_model.output))
model.compile(loss='categorical_crossentropy', optimizer=optimizers.Adam(learning_rate=0.0001,
```

```
beta_1=0.9, beta_2=0.999, epsilon=1e-05, amsgrad=False),
              metrics=['accuracy'])
model.summary()
steps_per_epoch = train_generator.n // batch_size
validation_steps = validation_generator.n // batch_size
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 208, 176, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 208, 176, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 208, 176, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 104, 88, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 104, 88, 128)      73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 104, 88, 128)      147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 52, 44, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 52, 44, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 52, 44, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 52, 44, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 26, 22, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 26, 22, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 26, 22, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 26, 22, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 13, 11, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 13, 11, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 13, 11, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 13, 11, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 5, 512)         0
_________________________________________________________________
sequential_1 (Sequential)    (None, 4)                 3841254
=================================================================
Total params: 18,555,942
Trainable params: 18,555,942
Non-trainable params: 0
_________________________________________________________________
```

In [4]:

```
model.fit_generator(train_generator,
        epochs=epoch,
        validation_data=validation_generator,
        verbose=1,
        steps_per_epoch=steps_per_epoch,
        validation_steps=validation_steps
        )
```

```
Epoch 1/13
420/420 [==============================] - 231s 551ms/step - loss: 1.1421 - accuracy: 0.4774 - val
_loss: 0.9145 - val_accuracy: 0.5487
Epoch 2/13
420/420 [==============================] - 210s 500ms/step - loss: 0.8866 - accuracy: 0.5877 - val
_loss: 1.3873 - val_accuracy: 0.6003
Epoch 3/13
420/420 [==============================] - 233s 556ms/step - loss: 0.7629 - accuracy: 0.6382 - val
_loss: 0.6445 - val_accuracy: 0.6019
Epoch 4/13
```

```
420/420 [==============================] - 264s 630ms/step - loss: 0.6504 - accuracy: 0.6912 - val
_loss: 1.1047 - val_accuracy: 0.6082
Epoch 5/13
420/420 [==============================] - 271s 644ms/step - loss: 0.5715 - accuracy: 0.7485 - val
_loss: 0.5675 - val_accuracy: 0.6414
Epoch 6/13
420/420 [==============================] - 269s 642ms/step - loss: 0.4776 - accuracy: 0.7941 - val
_loss: 1.8730 - val_accuracy: 0.6414
Epoch 7/13
420/420 [==============================] - 245s 584ms/step - loss: 0.3844 - accuracy: 0.8403 - val
_loss: 1.4435 - val_accuracy: 0.6635
Epoch 8/13
420/420 [==============================] - 219s 522ms/step - loss: 0.2834 - accuracy: 0.8877 - val
_loss: 3.6234 - val_accuracy: 0.6390
Epoch 9/13
420/420 [==============================] - 246s 586ms/step - loss: 0.2033 - accuracy: 0.9221 - val
_loss: 1.0489 - val_accuracy: 0.6675
Epoch 10/13
420/420 [==============================] - 260s 619ms/step - loss: 0.1671 - accuracy: 0.9397 - val
_loss: 2.8138 - val_accuracy: 0.6840
Epoch 11/13
420/420 [==============================] - 245s 583ms/step - loss: 0.1510 - accuracy: 0.9419 - val
_loss: 1.3946 - val_accuracy: 0.6651
Epoch 12/13
420/420 [==============================] - 231s 550ms/step - loss: 0.1194 - accuracy: 0.9564 - val
_loss: 2.6971 - val_accuracy: 0.6825
Epoch 13/13
420/420 [==============================] - 232s 552ms/step - loss: 0.0962 - accuracy: 0.9639 - val
_loss: 0.9044 - val_accuracy: 0.7441
```

Out[4]:

```
<keras.callbacks.callbacks.History at 0x1f0970bb588>
```

In [5]:

```
model.save("model1.h5")
```

In [ ]:

```
```