TEAM ID:PNT2022TMID14840

MODEL BUILDING

1. Importing The Model Building Libraries

In [5]:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

2. Loading The Model

In [6]:

```
IMAGE_SIZE = [224, 224]

train_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training'
valid_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/validation'
```

In [7]:

```
vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

3. Adding Flatten Layer

In [8]:

```
for layer in vgg16.layers:
    layer.trainable = False
```

In [11]:

```
folders = glob('/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/*')
```

In [12]:

```
folders
```

Out[12]:

```
['/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/03-severe',
 '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/02-moderate',
 '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/training/01-minor']
```

In [13]:

```
x = Flatten()(vgg16.output)
```

In [14]:

```
len(folders)
```

Out[14]:

```
3
```

4. Adding Output Layer

In [15]:

```
prediction = Dense(len(folders), activation='softmax')(x)
```

5. Creating A Model Object

In [16]:

```
model = Model(inputs=vgg16.input, outputs=prediction)
```

In [17]:

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 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_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

flatten (Flatten)           (None, 25088)             0

dense (Dense)               (None, 3)                 75267

=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
_________________________________________________________________
```

6. Configure The Learning Process

In [18]:

```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

7. Train The Model

In [19]:

```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=25,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_g
enerator` is deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
```

```
Epoch 1/25
98/98 [==============================] - 606s 6s/step - loss: 1.1697 - accuracy: 0.5608 -
val_loss: 0.9855 - val_accuracy: 0.6140
Epoch 2/25
98/98 [==============================] - 596s 6s/step - loss: 0.7030 - accuracy: 0.7099 -
val_loss: 0.9670 - val_accuracy: 0.6199
Epoch 3/25
98/98 [==============================] - 594s 6s/step - loss: 0.4431 - accuracy: 0.8202 -
val_loss: 1.0758 - val_accuracy: 0.5965
Epoch 4/25
98/98 [==============================] - 592s 6s/step - loss: 0.3887 - accuracy: 0.8570 -
val_loss: 1.0519 - val_accuracy: 0.6257
Epoch 5/25
98/98 [==============================] - 592s 6s/step - loss: 0.3058 - accuracy: 0.8856 -
val_loss: 1.5903 - val_accuracy: 0.6140
Epoch 6/25
98/98 [==============================] - 596s 6s/step - loss: 0.2978 - accuracy: 0.9019 -
val_loss: 1.1763 - val_accuracy: 0.6140
Epoch 7/25
98/98 [==============================] - 598s 6s/step - loss: 0.2060 - accuracy: 0.9295 -
```

```
Epoch 5/25
98/98 [==============================] - 592s 6s/step - loss: 0.3058 - accuracy: 0.8856 -
val_loss: 1.5903 - val_accuracy: 0.6140
Epoch 6/25
98/98 [==============================] - 596s 6s/step - loss: 0.2978 - accuracy: 0.9019 -
val_loss: 1.1763 - val_accuracy: 0.6140
Epoch 7/25
98/98 [==============================] - 598s 6s/step - loss: 0.2060 - accuracy: 0.9295 -
val_loss: 1.2846 - val_accuracy: 0.6082
Epoch 8/25
98/98 [==============================] - 596s 6s/step - loss: 0.1685 - accuracy: 0.9387 -
val_loss: 1.1337 - val_accuracy: 0.6023
Epoch 9/25
98/98 [==============================] - 595s 6s/step - loss: 0.1926 - accuracy: 0.9305 -
val_loss: 1.1559 - val_accuracy: 0.6725
Epoch 10/25
98/98 [==============================] - 594s 6s/step - loss: 0.1206 - accuracy: 0.9653 -
val_loss: 1.2013 - val_accuracy: 0.6433
Epoch 11/25
98/98 [==============================] - 595s 6s/step - loss: 0.1151 - accuracy: 0.9663 -
val_loss: 1.2582 - val_accuracy: 0.6023
Epoch 12/25
98/98 [==============================] - 595s 6s/step - loss: 0.0615 - accuracy: 0.9857 -
val_loss: 1.1696 - val_accuracy: 0.6608
Epoch 13/25
98/98 [==============================] - 597s 6s/step - loss: 0.0659 - accuracy: 0.9837 -
val_loss: 1.1735 - val_accuracy: 0.6374
Epoch 14/25
98/98 [==============================] - 597s 6s/step - loss: 0.0417 - accuracy: 0.9939 -
val_loss: 1.1479 - val_accuracy: 0.6433
Epoch 15/25
98/98 [==============================] - 597s 6s/step - loss: 0.0504 - accuracy: 0.9898 -
```

8. Save The Model

In [28]:

```
from tensorflow.keras.models import load_model

model.save('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimato
For Insurance Companies/Model/level.h5')
```

9. Test The Model

In [29]:

```
from tensorflow.keras.models import load_model
import cv2
from skimage.transform import resize
```

In [31]:

```
model = load_model('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost E
stimator For Insurance Companies/Model/level.h5')
```

In [25]:

```
def detect(frame):
  img = cv2.resize(frame,(224,224))
  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

  if(np.max(img)>1):
    img = img/255.0
  img = np.array([img])
  prediction = model.predict(img)
  label = ["minor","moderate","severe"]
  preds = label[np.argmax(prediction)]
  return preds
```

In [32]:

```
import numpy as np
```

In [33]:

```
data = "/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/level/va
lidation/01-minor/0008.jpeg"
image = cv2.imread(data)
print(detect(image))
```