MODEL BUILDING

1. Importing The Model Building Libraries

In [1]:

```
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 [2]:

```
IMAGE_SIZE = [224, 224]

train_path = '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment
& Cost Estimator For Insurance Companies/Dataset/body/training'
valid_path = '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment
& Cost Estimator For Insurance Companies/Dataset/body/validation'
```

In [ ]:

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

3. Adding Flatten Layer

In [3]:

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
 in
----> 1 for layer in vgg16.layers:
      2     layer.trainable = False

NameError: name 'vgg16' is not defined
```

In [4]:

```
folders = glob('/content/drive/MyDrive/Intelligent Vehicle Damage
Assessment & Cost Estimator For Insurance
Companies/Dataset/body/training/*')
```

In [ ]:

```
folders
```

Out[ ]:

```
['/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estim
ator For Insurance Companies/Dataset/body/training/02-side',
 '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estim
ator For Insurance Companies/Dataset/body/training/00-front',
 '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estim
ator For Insurance Companies/Dataset/body/training/01-rear']
```

In [ ]:

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

In [ ]:

```
len(folders)
```

Out[ ]:

```
3
```

4. Adding Output Layer

In [ ]:

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

5. Creating A Model Object

In [ ]:

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

In [ ]:

```
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_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 [ ]:

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

7. Train The Model

In [ ]:

```
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_generator` is deprecated and will be removed in a future versi
on. Please use `Model.fit`, which supports generators.

Epoch 1/25
98/98 [==============================] - 606s 6s/step - loss: 1.2827 - accu
racy: 0.5649 - val_loss: 0.8292 - val_accuracy: 0.7076
Epoch 2/25
98/98 [==============================] - 601s 6s/step - loss: 0.6301 - accu
racy: 0.7467 - val_loss: 1.2482 - val_accuracy: 0.5965
Epoch 3/25
98/98 [==============================] - 601s 6s/step - loss: 0.5073 - accu
racy: 0.8039 - val_loss: 0.8174 - val_accuracy: 0.7193
Epoch 4/25
98/98 [==============================] - 601s 6s/step - loss: 0.3564 - accu
racy: 0.8621 - val_loss: 0.9245 - val_accuracy: 0.6608
Epoch 5/25
98/98 [==============================] - 599s 6s/step - loss: 0.2951 - accu
racy: 0.8917 - val_loss: 1.9934 - val_accuracy: 0.5906
Epoch 6/25
98/98 [==============================] - 638s 7s/step - loss: 0.2557 - accu
racy: 0.9152 - val_loss: 0.9176 - val_accuracy: 0.6842
Epoch 7/25
98/98 [==============================] - 607s 6s/step - loss: 0.2083 - accu
racy: 0.9367 - val_loss: 0.9594 - val_accuracy: 0.7018
Epoch 8/25
98/98 [==============================] - 600s 6s/step - loss: 0.2184 - accu
racy: 0.9122 - val_loss: 1.0329 - val_accuracy: 0.6784
Epoch 9/25
```

```
98/98 [==============================] - 602s 6s/step - loss: 0.1320 - accu
racy: 0.9581 - val_loss: 1.0539 - val_accuracy: 0.7135
Epoch 10/25
98/98 [==============================] - 599s 6s/step - loss: 0.1131 - accu
racy: 0.9622 - val_loss: 1.2113 - val_accuracy: 0.6842
Epoch 11/25
98/98 [==============================] - 597s 6s/step - loss: 0.1001 - accu
racy: 0.9745 - val_loss: 0.9917 - val_accuracy: 0.7018
Epoch 12/25
98/98 [==============================] - 598s 6s/step - loss: 0.0954 - accu
racy: 0.9745 - val_loss: 1.0601 - val_accuracy: 0.7018
Epoch 13/25
98/98 [==============================] - 594s 6s/step - loss: 0.0695 - accu
racy: 0.9816 - val_loss: 1.3700 - val_accuracy: 0.6433
Epoch 14/25
98/98 [==============================] - 599s 6s/step - loss: 0.1414 - accu
racy: 0.9653 - val_loss: 1.1607 - val_accuracy: 0.6667
Epoch 15/25
98/98 [==============================] - 600s 6s/step - loss: 0.0905 - accu
racy: 0.9796 - val_loss: 1.4014 - val_accuracy: 0.6667
Epoch 16/25
98/98 [==============================] - 601s 6s/step - loss: 0.0797 - accu
racy: 0.9775 - val_loss: 1.6741 - val_accuracy: 0.6491
Epoch 17/25
98/98 [==============================] - 602s 6s/step - loss: 0.1042 - accu
racy: 0.9745 - val_loss: 1.2824 - val_accuracy: 0.6959
Epoch 18/25
98/98 [==============================] - 600s 6s/step - loss: 0.0831 - accu
racy: 0.9785 - val_loss: 1.1667 - val_accuracy: 0.6901
Epoch 19/25
98/98 [==============================] - 603s 6s/step - loss: 0.0826 - accu
racy: 0.9704 - val_loss: 1.3747 - val_accuracy: 0.6374
Epoch 20/25
98/98 [==============================] - 600s 6s/step - loss: 0.0536 - accu
racy: 0.9837 - val_loss: 1.2074 - val_accuracy: 0.6550
Epoch 21/25
98/98 [==============================] - 597s 6s/step - loss: 0.0716 - accu
racy: 0.9796 - val_loss: 1.5491 - val_accuracy: 0.6725
Epoch 22/25
98/98 [==============================] - 599s 6s/step - loss: 0.0457 - accu
racy: 0.9918 - val_loss: 1.2930 - val_accuracy: 0.7135
Epoch 23/25
98/98 [==============================] - 601s 6s/step - loss: 0.0526 - accu
racy: 0.9928 - val_loss: 1.2576 - val_accuracy: 0.6959
Epoch 24/25
98/98 [==============================] - 601s 6s/step - loss: 0.0421 - accu
racy: 0.9908 - val_loss: 1.3347 - val_accuracy: 0.7193
Epoch 25/25
98/98 [==============================] - 597s 6s/step - loss: 0.0597 - accu
racy: 0.9826 - val_loss: 1.4728 - val_accuracy: 0.6725
```

8. Save The Model

In [ ]:

```
from tensorflow.keras.models import load_model

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

9. Test The Model

In [ ]:

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

In [ ]:

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

In [ ]:

```
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 = ["front","rear","side"]
  preds = label[np.argmax(prediction)]
  return preds
```

In [ ]:

```
import numpy as np
```

In [ ]:

```
data = "/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost
Estimator For Insurance Companies/Dataset/body/training/00-front/0005.JPEG"
image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 1s 638ms/step
front
```