MODEL BUILDING

Body:

1. Importing The Model Building Libraries

```
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

```
IMAGE_SIZE = [224, 224]
train_path = '/content/drive/MyDrive/body/training' valid_path =
'/content/drive/MyDrive/body/validation' vgg16 = VGG16(input_shape=IMAGE_SIZE + [3],
weights='imagenet', include_top=False)
```

```
Downloading data from
https://storage.googleapis.com/tensorflow/kerasapplications/vgg16/vgg16_weights_tf_dim_ordering_
tf_kernels_notop.h5 58889256/58889256 [==============================] - 0s 0us/step
```

3. Adding Flatten Layer

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

```
folders = glob('/content/drive/MyDrive/body/training/*') folders
```

```
['/content/drive/MyDrive/body/training/02-side',
 '/content/drive/MyDrive/body/training/00-front',  '/content/drive/MyDrive/body/training/01-rear']
```

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

```
3
```

4. Adding Output Layer

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

5. Creating A Model Object

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

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

7. Train The Model

8. Save The Model

```
from tensorflow.keras.models import load_model
model.save('/content/drive/MyDrive/ibm project/Intelligent Vehicle Damage Assessment & Cost Estimator/MODEL/BODY.h5')
```

9. Test The Model

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

model = load_model('/content/drive/MyDrive/ibm project/Intelligent Vehicle Damage Assessment & Cost Estimator/MODEL/BODY.h5')

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

data = "/content/drive/MyDrive/body/training/00-front/0007.JPEG" image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 1s 812ms/step front
```

Level:

1. Importing The Model Building Libraries

```
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

```
IMAGE_SIZE = [224, 224]
train_path = '/content/drive/MyDrive/level/training' valid_path =
'/content/drive/MyDrive/level/validation'
```

3. Adding Flatten Layer

```
vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
Downloading data from
https://storage.googleapis.com/tensorflow/kerasapplications/vgg16/vgg16_weights_tf_dim_ordering_
tf_kernels_notop.h5 58889256/58889256 [==============================] - 2s 0us/step
```

```
for layer in vgg16.layers:layer.trainable = False folders =
glob('/content/drive/MyDrive/level/training/*') folders
```

```
['/content/drive/MyDrive/level/training/02-moderate',
 '/content/drive/MyDrive/level/training/03-severe',  '/content/drive/MyDrive/level/training/01-minor']
```

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

```
3
```

4. Adding Output Layer

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

5. Creating A Model Object

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

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

7. Train The Model

```
r = model.fit_generator( training_set,
validation_data=test_set, epochs=5,
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 version. Please use `Model.fit`, which supports generators.

Epoch 1/5
98/98 [==============================] - 407s 4s/step - loss: 1.2409 -
accuracy: 0.5628 - val_loss: 1.2019 - val_accuracy: 0.5614 Epoch 2/5
98/98 [==============================] - 18s 179ms/step - loss: 0.7316
- accuracy: 0.7191 - val_loss: 0.9586 - val_accuracy: 0.6082
Epoch 3/5
98/98 [==============================] - 16s 164ms/step - loss: 0.5469
- accuracy: 0.7957 - val_loss: 1.0207 - val_accuracy: 0.6140
Epoch 4/5
98/98 [==============================] - 16s 167ms/step - loss: 0.4278
- accuracy: 0.8223 - val_loss: 1.6515 - val_accuracy: 0.5965
Epoch 5/5
98/98 [==============================] - 17s 177ms/step - loss: 0.4449
- accuracy: 0.8284 - val_loss: 1.2299 - val_accuracy: 0.6199

8. Save The Model

```
from tensorflow.keras.models import load_model
model.save('/content/drive/MyDrive/ibm project/Intelligent Vehicle Damage Assessment & Cost Estimator/MODEL/LEVEL.h5')
```

9. Test The Model

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

model = load_model('/content/drive/MyDrive/ibm project/Intelligent Vehicle Damage Assessment & Cost Estimator/MODEL/LEVEL.h5')

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

data = "/content/drive/MyDrive/level/training/01-minor/0007.JPEG" image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 0s 157ms/step minor
```