**FOR BODY DAMAGE**

**IMAGE PRE PROCESSING**

**1. Import The ImageDataGenerator Library**

In [ ]:

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

**2. Configure ImageDataGenerator Class**

In [ ]:

train\_datagen **=** ImageDataGenerator(rescale **=** 1.**/**255,

shear\_range **=** 0.1,

zoom\_range **=** 0.1,

horizontal\_flip **=** **True**)

test\_datagen **=** ImageDataGenerator(rescale **=** 1.**/**255)

**3. Apply ImageDataGenerator Functionality To Trainset And Testset**

In [ ]:

training\_set **=** train\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/body/training',

target\_size **=** (224, 224),

batch\_size **=** 10,

class\_mode **=** 'categorical')

test\_set **=** test\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/body/validation',

target\_size **=** (224, 224),

batch\_size **=** 10,

class\_mode **=** 'categorical')

Found 979 images belonging to 3 classes.

Found 171 images belonging to 3 classes.

**MODEL BUILDING**

**1. Importing The Model Building Libraries**

In [ ]:

**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 [ ]:

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 [ ]:

**for** layer **in** vgg16**.**layers:

layer**.**trainable **=** **False**

In [ ]:

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 Estimator For Insurance Companies/Dataset/body/training/02-side',

'/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/body/training/00-front',

'/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator 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 version. Please use `Model.fit`, which supports generators.

Epoch 1/25

98/98 [==============================] - 606s 6s/step - loss: 1.2827 - accuracy: 0.5649 - val\_loss: 0.8292 - val\_accuracy: 0.7076

Epoch 2/25

98/98 [==============================] - 601s 6s/step - loss: 0.6301 - accuracy: 0.7467 - val\_loss: 1.2482 - val\_accuracy: 0.5965

Epoch 3/25

98/98 [==============================] - 601s 6s/step - loss: 0.5073 - accuracy: 0.8039 - val\_loss: 0.8174 - val\_accuracy: 0.7193

Epoch 4/25

98/98 [==============================] - 601s 6s/step - loss: 0.3564 - accuracy: 0.8621 - val\_loss: 0.9245 - val\_accuracy: 0.6608

Epoch 5/25

98/98 [==============================] - 599s 6s/step - loss: 0.2951 - accuracy: 0.8917 - val\_loss: 1.9934 - val\_accuracy: 0.5906

Epoch 6/25

98/98 [==============================] - 638s 7s/step - loss: 0.2557 - accuracy: 0.9152 - val\_loss: 0.9176 - val\_accuracy: 0.6842

Epoch 7/25

98/98 [==============================] - 607s 6s/step - loss: 0.2083 - accuracy: 0.9367 - val\_loss: 0.9594 - val\_accuracy: 0.7018

Epoch 8/25

98/98 [==============================] - 600s 6s/step - loss: 0.2184 - accuracy: 0.9122 - val\_loss: 1.0329 - val\_accuracy: 0.6784

Epoch 9/25

98/98 [==============================] - 602s 6s/step - loss: 0.1320 - accuracy: 0.9581 - val\_loss: 1.0539 - val\_accuracy: 0.7135

Epoch 10/25

98/98 [==============================] - 599s 6s/step - loss: 0.1131 - accuracy: 0.9622 - val\_loss: 1.2113 - val\_accuracy: 0.6842

Epoch 11/25

98/98 [==============================] - 597s 6s/step - loss: 0.1001 - accuracy: 0.9745 - val\_loss: 0.9917 - val\_accuracy: 0.7018

Epoch 12/25

98/98 [==============================] - 598s 6s/step - loss: 0.0954 - accuracy: 0.9745 - val\_loss: 1.0601 - val\_accuracy: 0.7018

Epoch 13/25

98/98 [==============================] - 594s 6s/step - loss: 0.0695 - accuracy: 0.9816 - val\_loss: 1.3700 - val\_accuracy: 0.6433

Epoch 14/25

98/98 [==============================] - 599s 6s/step - loss: 0.1414 - accuracy: 0.9653 - val\_loss: 1.1607 - val\_accuracy: 0.6667

Epoch 15/25

98/98 [==============================] - 600s 6s/step - loss: 0.0905 - accuracy: 0.9796 - val\_loss: 1.4014 - val\_accuracy: 0.6667

Epoch 16/25

98/98 [==============================] - 601s 6s/step - loss: 0.0797 - accuracy: 0.9775 - val\_loss: 1.6741 - val\_accuracy: 0.6491

Epoch 17/25

98/98 [==============================] - 602s 6s/step - loss: 0.1042 - accuracy: 0.9745 - val\_loss: 1.2824 - val\_accuracy: 0.6959

Epoch 18/25

98/98 [==============================] - 600s 6s/step - loss: 0.0831 - accuracy: 0.9785 - val\_loss: 1.1667 - val\_accuracy: 0.6901

Epoch 19/25

98/98 [==============================] - 603s 6s/step - loss: 0.0826 - accuracy: 0.9704 - val\_loss: 1.3747 - val\_accuracy: 0.6374

Epoch 20/25

98/98 [==============================] - 600s 6s/step - loss: 0.0536 - accuracy: 0.9837 - val\_loss: 1.2074 - val\_accuracy: 0.6550

Epoch 21/25

98/98 [==============================] - 597s 6s/step - loss: 0.0716 - accuracy: 0.9796 - val\_loss: 1.5491 - val\_accuracy: 0.6725

Epoch 22/25

98/98 [==============================] - 599s 6s/step - loss: 0.0457 - accuracy: 0.9918 - val\_loss: 1.2930 - val\_accuracy: 0.7135

Epoch 23/25

98/98 [==============================] - 601s 6s/step - loss: 0.0526 - accuracy: 0.9928 - val\_loss: 1.2576 - val\_accuracy: 0.6959

Epoch 24/25

98/98 [==============================] - 601s 6s/step - loss: 0.0421 - accuracy: 0.9908 - val\_loss: 1.3347 - val\_accuracy: 0.7193

Epoch 25/25

98/98 [==============================] - 597s 6s/step - loss: 0.0597 - accuracy: 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

**FOR LEVEL DAMAGE**

**IMAGE PRE PROCESSING**

**1. Import The ImageDataGenerator Library**

In [2]:

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

**2. Configure ImageDataGenerator Class**

In [3]:

train\_datagen **=** ImageDataGenerator(rescale **=** 1.**/**255,

shear\_range **=** 0.1,

zoom\_range **=** 0.1,

horizontal\_flip **=** **True**)

test\_datagen **=** ImageDataGenerator(rescale **=** 1.**/**255)

**3. Apply ImageDataGenerator Functionality To Trainset And Testset**

In [4]:

training\_set **=** train\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/level/training',

target\_size **=** (224, 224),

batch\_size **=** 10,

class\_mode **=** 'categorical')

test\_set **=** test\_datagen**.**flow\_from\_directory('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/level/validation',

target\_size **=** (224, 224),

batch\_size **=** 10,

class\_mode **=** 'categorical')

Found 979 images belonging to 3 classes.

Found 171 images belonging to 3 classes.

**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/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/level/training'

valid\_path **=** '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/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 [9]:

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

In [10]:

folders

Out[10]:

['/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/level/training/02-moderate',

'/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/level/training/03-severe',

'/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/level/training/01-minor']

In [11]:

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

In [12]:

len(folders)

Out[12]:

3

**4. Adding Output Layer**

In [13]:

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

**5. Creating A Model Object**

In [14]:

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

In [15]:

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 [16]:

model**.**compile(

loss**=**'categorical\_crossentropy',

optimizer**=**'adam',

metrics**=**['accuracy']

)

**7. Train The Model**

In [17]:

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 version. Please use `Model.fit`, which supports generators.

Epoch 1/25

98/98 [==============================] - 615s 6s/step - loss: 1.2465 - accuracy: 0.5516 - val\_loss: 1.0659 - val\_accuracy: 0.5731

Epoch 2/25

98/98 [==============================] - 604s 6s/step - loss: 0.6654 - accuracy: 0.7549 - val\_loss: 1.0368 - val\_accuracy: 0.6316

Epoch 3/25

98/98 [==============================] - 604s 6s/step - loss: 0.5950 - accuracy: 0.7630 - val\_loss: 1.1309 - val\_accuracy: 0.6257

Epoch 4/25

98/98 [==============================] - 601s 6s/step - loss: 0.4964 - accuracy: 0.8069 - val\_loss: 1.1262 - val\_accuracy: 0.6082

Epoch 5/25

98/98 [==============================] - 603s 6s/step - loss: 0.3559 - accuracy: 0.8672 - val\_loss: 1.1408 - val\_accuracy: 0.6316

Epoch 6/25

98/98 [==============================] - 604s 6s/step - loss: 0.2425 - accuracy: 0.9152 - val\_loss: 1.1566 - val\_accuracy: 0.5789

Epoch 7/25

98/98 [==============================] - 604s 6s/step - loss: 0.1964 - accuracy: 0.9367 - val\_loss: 1.1200 - val\_accuracy: 0.6199

Epoch 8/25

98/98 [==============================] - 598s 6s/step - loss: 0.2119 - accuracy: 0.9203 - val\_loss: 1.1181 - val\_accuracy: 0.6316

Epoch 9/25

98/98 [==============================] - 597s 6s/step - loss: 0.1111 - accuracy: 0.9622 - val\_loss: 1.3554 - val\_accuracy: 0.5614

Epoch 10/25

98/98 [==============================] - 595s 6s/step - loss: 0.1394 - accuracy: 0.9438 - val\_loss: 1.2256 - val\_accuracy: 0.6082

Epoch 11/25

98/98 [==============================] - 598s 6s/step - loss: 0.1167 - accuracy: 0.9602 - val\_loss: 1.3020 - val\_accuracy: 0.6374

Epoch 12/25

98/98 [==============================] - 598s 6s/step - loss: 0.0823 - accuracy: 0.9755 - val\_loss: 1.3000 - val\_accuracy: 0.6550

Epoch 13/25

98/98 [==============================] - 602s 6s/step - loss: 0.1062 - accuracy: 0.9632 - val\_loss: 1.2962 - val\_accuracy: 0.6433

Epoch 14/25

98/98 [==============================] - 599s 6s/step - loss: 0.0717 - accuracy: 0.9775 - val\_loss: 1.3089 - val\_accuracy: 0.6491

Epoch 15/25

98/98 [==============================] - 598s 6s/step - loss: 0.0692 - accuracy: 0.9826 - val\_loss: 1.2885 - val\_accuracy: 0.6023

Epoch 16/25

98/98 [==============================] - 595s 6s/step - loss: 0.0449 - accuracy: 0.9898 - val\_loss: 1.7932 - val\_accuracy: 0.5673

Epoch 17/25

98/98 [==============================] - 609s 6s/step - loss: 0.0522 - accuracy: 0.9867 - val\_loss: 1.2697 - val\_accuracy: 0.6433

Epoch 18/25

98/98 [==============================] - 607s 6s/step - loss: 0.0386 - accuracy: 0.9969 - val\_loss: 1.5100 - val\_accuracy: 0.6023

Epoch 19/25

98/98 [==============================] - 595s 6s/step - loss: 0.0381 - accuracy: 0.9939 - val\_loss: 1.2199 - val\_accuracy: 0.6784

Epoch 20/25

98/98 [==============================] - 596s 6s/step - loss: 0.0196 - accuracy: 1.0000 - val\_loss: 1.2907 - val\_accuracy: 0.6433

Epoch 21/25

98/98 [==============================] - 597s 6s/step - loss: 0.0394 - accuracy: 0.9928 - val\_loss: 1.2678 - val\_accuracy: 0.6491

Epoch 22/25

98/98 [==============================] - 595s 6s/step - loss: 0.0377 - accuracy: 0.9908 - val\_loss: 1.4709 - val\_accuracy: 0.6316

Epoch 23/25

98/98 [==============================] - 595s 6s/step - loss: 0.0387 - accuracy: 0.9918 - val\_loss: 1.3320 - val\_accuracy: 0.6257

Epoch 24/25

98/98 [==============================] - 596s 6s/step - loss: 0.0279 - accuracy: 0.9949 - val\_loss: 1.6355 - val\_accuracy: 0.6433

Epoch 25/25

98/98 [==============================] - 603s 6s/step - loss: 0.0271 - accuracy: 0.9939 - val\_loss: 1.3182 - val\_accuracy: 0.6608

**8. Save The Model**

In [18]:

**from** tensorflow.keras.models **import** load\_model

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

**9. Test The Model**

In [19]:

**from** tensorflow.keras.models **import** load\_model

**import** cv2

**from** skimage.transform **import** resize

In [20]:

model **=** load\_model('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Model/level.h5')

In [21]:

**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 [22]:

**import** numpy **as** np

In [23]:

data **=** "/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For Insurance Companies/Dataset/level/validation/01-minor/0010.JPEG"

image **=** cv2**.**imread(data)

print(detect(image))

1/1 [==============================] - 1s 728ms/step

minor