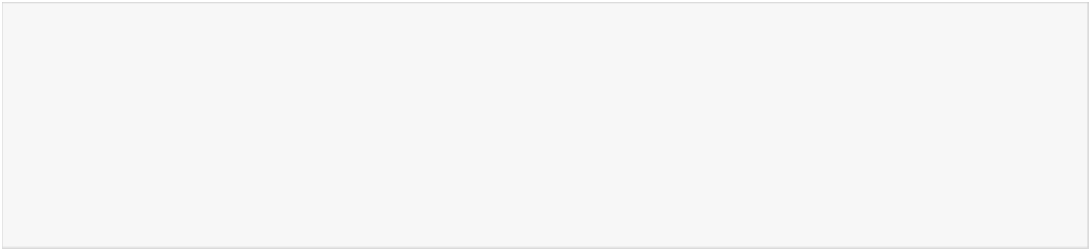
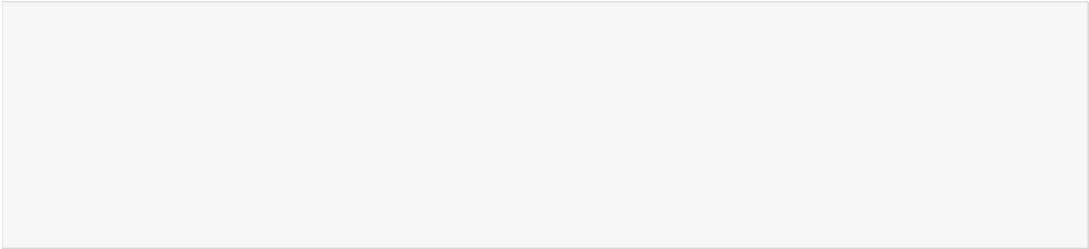
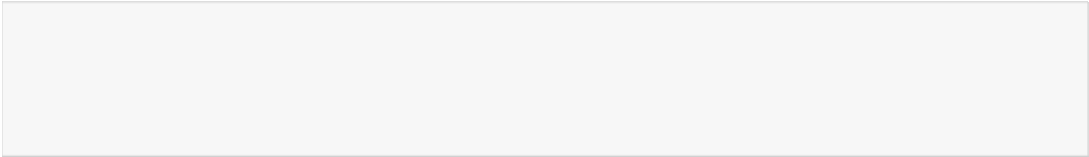
|  |  |
| --- | --- |
| **TEAM ID** | PNT2022TMID12053 |
| **DATE** | 17/10/2022 |
| **PROJECT NAME** | Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies |
| **TEAM MEMBERS** | Aravintan.T  Diwahar.S.J  Sharan.s  Sneha.Y |

# FOR BODY DAMAGE IMAGE PRE PROCESSING



**1. Import The ImageDataGenerator Library**

In [ ]:

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# :2. Configure ImageDataGenerator Class Image Data Augmentation

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/IBM - PROJECT/Da ta set/body-20221023T072112Z-001/body/training',

target\_size = (224, 224),

batch\_size = 10, class\_mode = 'categorical')

test\_set = test\_datagen.flow\_from\_directory('/content/drive/MyDrive/IBM - PROJECT/Data se t/body-20221023T072112Z-001/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

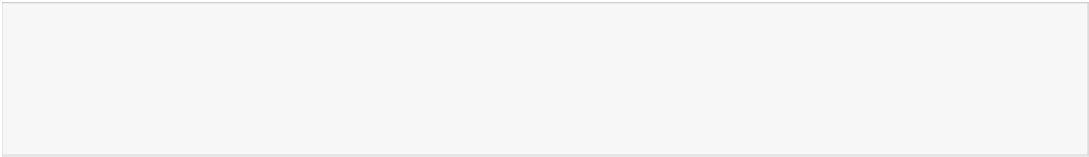
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load\_img from tensorflow.keras.models import Sequential

import numpy as np from glob import glob



# Loading The Model

In [ ]:



IMAGE\_SIZE = [224, 224]

train\_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/bod y/training'

valid\_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/bod y/validation'

In [ ]:



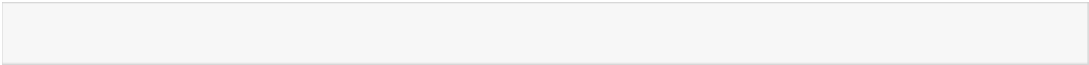
vgg16 = VGG16(input\_shape=IMAGE\_SIZE + [3], weights='imagenet', include\_top=False)

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

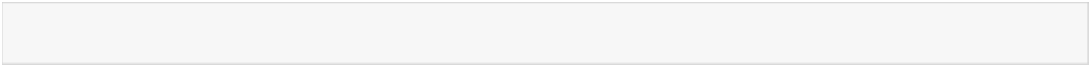
# Adding Flatten Layer

In [ ]:



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

In [ ]:



folders = glob('/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/b ody/training/\*')

In [ ]:



folders

Out[ ]:

['/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/0 2-side',

'/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/0 1-rear',

'/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/training/0 0-front']

In [ ]:



x = Flatten()(vgg16.output)

In [ ]:



len(folders)

Out[ ]:

3

## Adding Output Layer

In [ ]:



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

## 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 |

=================================================================

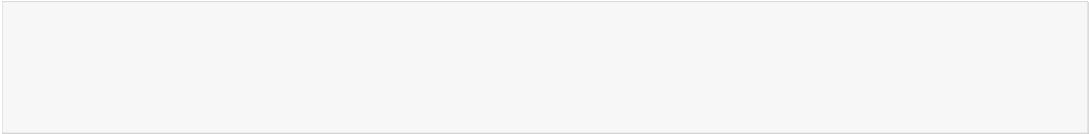
Total params: 14,789,955

Trainable params: 75,267

Non-trainable params: 14,714,688

## Configure The Learning Process

In [ ]:



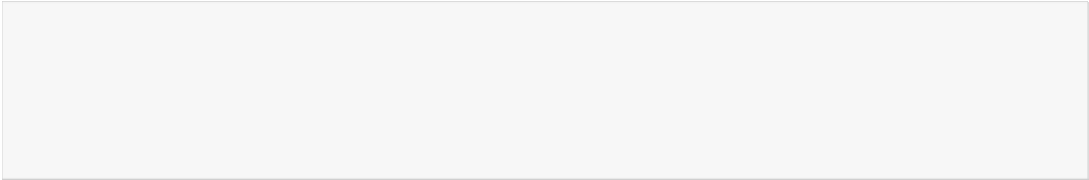
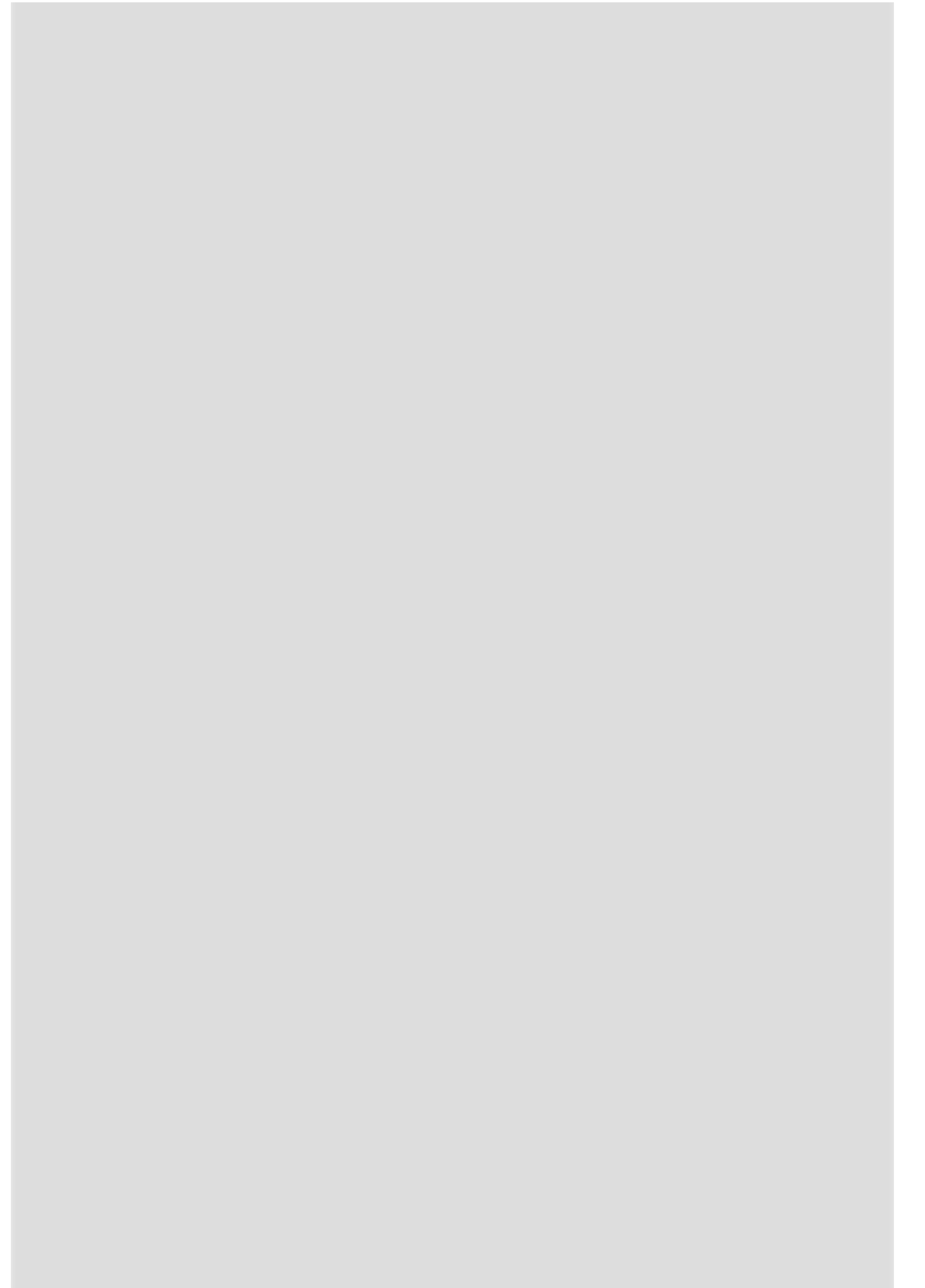
model.compile( loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy']

)

## 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\_g enerator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

Epoch 1/25

98/98 [==============================] - 560s 6s/step - loss: 1.2275 - accuracy: 0.5383 -

val\_loss: 0.8698 - val\_accuracy: 0.6608 Epoch 2/25

98/98 [==============================] - 584s 6s/step - loss: 0.7810 - accuracy: 0.7007 -

val\_loss: 0.8931 - val\_accuracy: 0.6491 Epoch 3/25

98/98 [==============================] - 538s 5s/step - loss: 0.4842 - accuracy: 0.8264 -

val\_loss: 0.8348 - val\_accuracy: 0.6842 Epoch 4/25

98/98 [==============================] - 537s 5s/step - loss: 0.3813 - accuracy: 0.8560 -

val\_loss: 0.9010 - val\_accuracy: 0.6901 Epoch 5/25

98/98 [==============================] - 537s 5s/step - loss: 0.2735 - accuracy: 0.8999 -

val\_loss: 1.0660 - val\_accuracy: 0.6901 Epoch 6/25

98/98 [==============================] - 538s 5s/step - loss: 0.2211 - accuracy: 0.9295 -

val\_loss: 1.0073 - val\_accuracy: 0.7076 Epoch 7/25

98/98 [==============================] - 536s 5s/step - loss: 0.2163 - accuracy: 0.9224 -

val\_loss: 0.9560 - val\_accuracy: 0.7251 Epoch 8/25

98/98 [==============================] - 538s 6s/step - loss: 0.1728 - accuracy: 0.9397 -

val\_loss: 1.0719 - val\_accuracy: 0.6491 Epoch 9/25

98/98 [==============================] - 540s 6s/step - loss: 0.1423 - accuracy: 0.9581 -

val\_loss: 1.0706 - val\_accuracy: 0.6901 Epoch 10/25

98/98 [==============================] - 539s 6s/step - loss: 0.1118 - accuracy: 0.9704 -

val\_loss: 1.1651 - val\_accuracy: 0.6842 Epoch 11/25

98/98 [==============================] - 538s 5s/step - loss: 0.0808 - accuracy: 0.9785 -

val\_loss: 1.1212 - val\_accuracy: 0.7076 Epoch 12/25

98/98 [==============================] - 549s 6s/step - loss: 0.0751 - accuracy: 0.9857 -

val\_loss: 1.1451 - val\_accuracy: 0.6842 Epoch 13/25

98/98 [==============================] - 555s 6s/step - loss: 0.0730 - accuracy: 0.9816 -

val\_loss: 1.0812 - val\_accuracy: 0.6842 Epoch 14/25

98/98 [==============================] - 535s 5s/step - loss: 0.1074 - accuracy: 0.9734 -

val\_loss: 1.2204 - val\_accuracy: 0.6842 Epoch 15/25

98/98 [==============================] - 539s 6s/step - loss: 0.0598 - accuracy: 0.9888 -

val\_loss: 1.6480 - val\_accuracy: 0.6316 Epoch 16/25

98/98 [==============================] - 543s 6s/step - loss: 0.0810 - accuracy: 0.9806 -

val\_loss: 1.2050 - val\_accuracy: 0.6901 Epoch 17/25

98/98 [==============================] - 541s 6s/step - loss: 0.1196 - accuracy: 0.9632 -

val\_loss: 1.3478 - val\_accuracy: 0.6374 Epoch 18/25

98/98 [==============================] - 543s 6s/step - loss: 0.0915 - accuracy: 0.9755 -

val\_loss: 1.2961 - val\_accuracy: 0.7018 Epoch 19/25

98/98 [==============================] - 544s 6s/step - loss: 0.0687 - accuracy: 0.9806 -

val\_loss: 1.2175 - val\_accuracy: 0.6842 Epoch 20/25

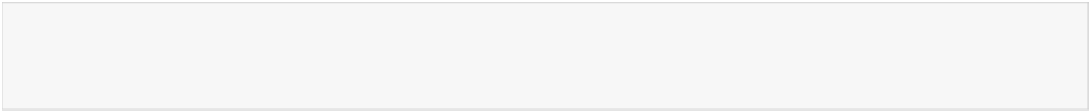
98/98 [==============================] - 546s 6s/step - loss: 0.0492 - accuracy: 0.9918 -



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| val\_loss: 1.3791 - val\_accuracy: 0.6784 |  | | | | | | |
| Epoch 21/25 |
| 98/98 [==============================] - | 543s | 6s/step | - loss: | 0.0674 | - accuracy: | 0.9847 | - |
| val\_loss: 1.5585 - val\_accuracy: 0.6433 |  |  |  |  |  |  |  |
| Epoch 22/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 537s | 5s/step | - loss: | 0.0740 | - accuracy: | 0.9775 | - |
| val\_loss: 1.7693 - val\_accuracy: 0.6550 |  |  |  |  |  |  |  |
| Epoch 23/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 538s | 6s/step | - loss: | 0.0822 | - accuracy: | 0.9765 | - |
| val\_loss: 1.9127 - val\_accuracy: 0.6374 |  |  |  |  |  |  |  |
| Epoch 24/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 541s | 6s/step | - loss: | 0.1048 | - accuracy: | 0.9653 | - |
| val\_loss: 1.5448 - val\_accuracy: 0.6316 |  |  |  |  |  |  |  |
| Epoch 25/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 544s | 6s/step | - loss: | 0.1373 | - accuracy: | 0.9551 | - |
| val\_loss: 1.4574 - val\_accuracy: 0.6842 |  |  |  |  |  |  |  |
| **8. Save The Model**  In [ ]: |  |  |  |  |  |  |  |



In [ ]:

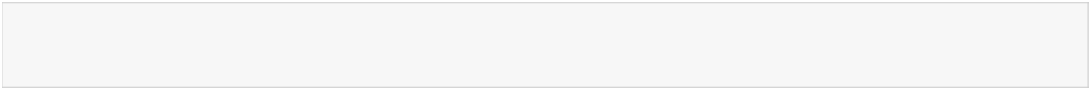


from tensorflow.keras.models import load\_model

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

# 9. Test The Model

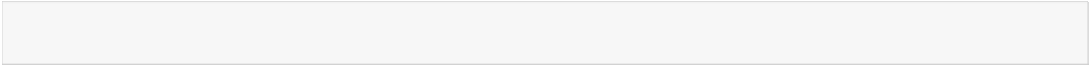
In [ ]:



from tensorflow.keras.models import load\_model import cv2

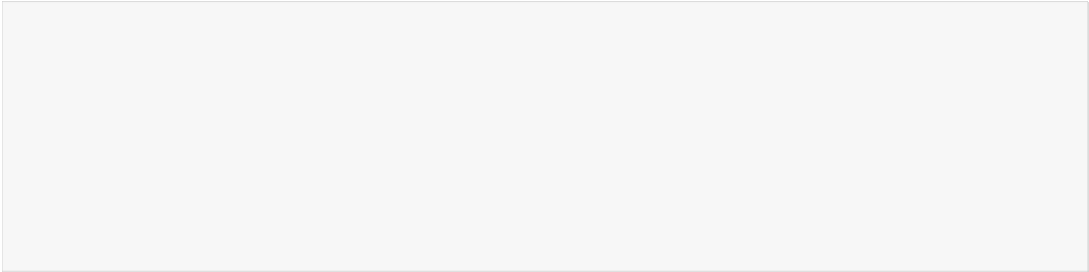
from skimage.transform import resize

In [ ]:



model = load\_model('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost E stimator 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"]

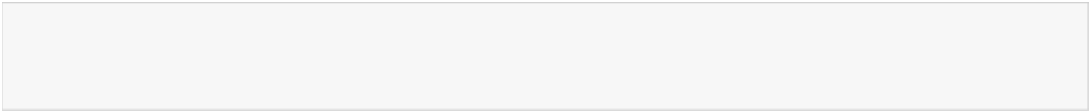
preds = label[np.argmax(prediction)] return preds

In [ ]:



import numpy as np

In [ ]:



data = "/content/drive/MyDrive/IBM - PROJECT/Data set/body-20221023T072112Z-001/body/trai ning/00-front/0008.jpeg"

image = cv2.imread(data) print(detect(image))

1/1 [==============================] - 0s 498ms/step

front

FOR LEVEL DAMAGE IMAGE PRE PROCESSING

1. Import The ImageDataGenerator Library

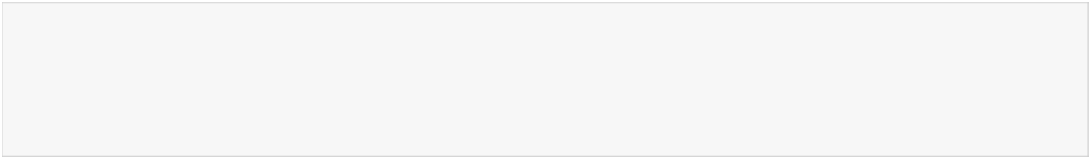
In [1]:



from tensorflow.keras.preprocessing.image import ImageDataGenerator

1. Configure ImageDataGenerator Class

In [2]:



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

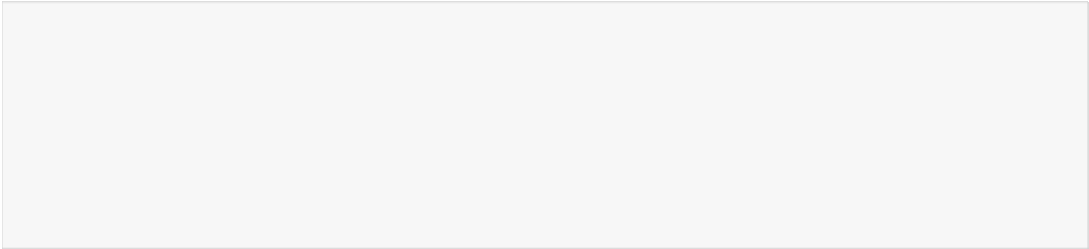
shear\_range = 0.1,

zoom\_range = 0.1, horizontal\_flip = True)

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

1. Apply ImageDataGenerator Functionality To Trainset And Testset

In [4]:



training\_set = train\_datagen.flow\_from\_directory('/content/drive/MyDrive/IBM - PROJECT/Da ta set/level-20221023T072121Z-001/level/training',

target\_size = (224, 224),

batch\_size = 10, class\_mode = 'categorical')

test\_set = test\_datagen.flow\_from\_directory('/content/drive/MyDrive/IBM - PROJECT/Data se t/level-20221023T072121Z-001/level/validation',

target\_size = (224, 224),

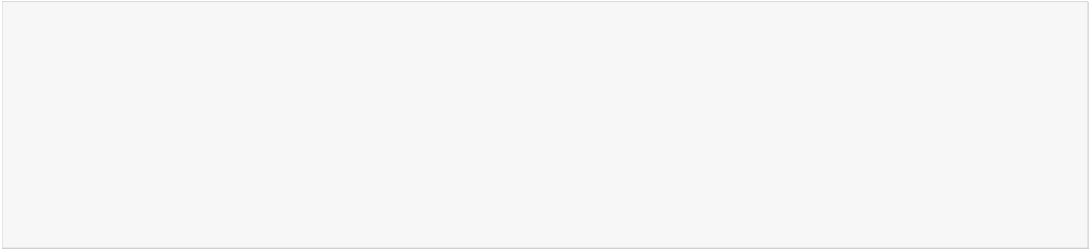
batch\_size = 10, class\_mode = 'categorical')

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

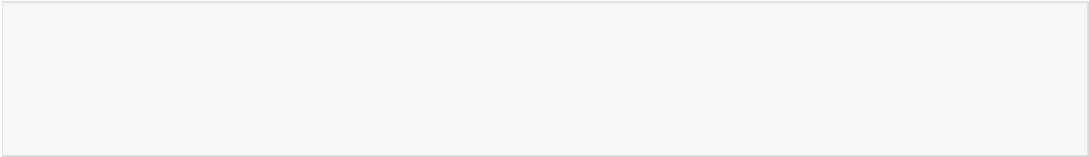
from tensorflow.keras.applications.vgg16 import VGG16 from tensorflow.keras.applications.vgg19 import VGG19 from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator,load\_img from tensorflow.keras.models import Sequential

import numpy as np from glob import glob

1. **Loading The Model**

In [6]:



IMAGE\_SIZE = [224, 224]

train\_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/le vel/training'

valid\_path = '/content/drive/MyDrive/IBM - PROJECT/Data set/level-20221023T072121Z-001/le vel/validation'

In [7]:



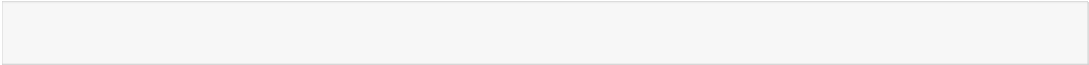
vgg16 = VGG16(input\_shape=IMAGE\_SIZE + [3], weights='imagenet', include\_top=False)

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

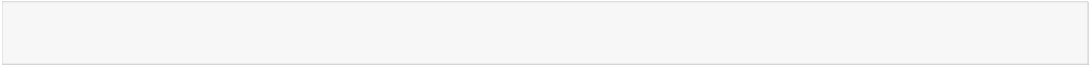
# Adding Flatten Layer

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

# Adding Output Layer

In [15]:



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

# 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\_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 |

=================================================================

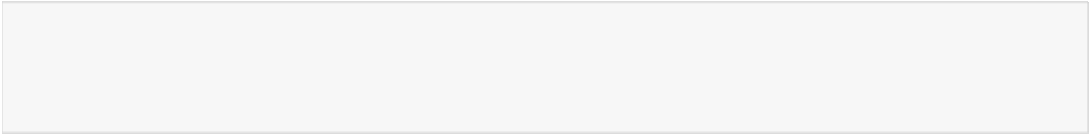
Total params: 14,789,955

Trainable params: 75,267

Non-trainable params: 14,714,688

# Configure The Learning Process

In [18]:



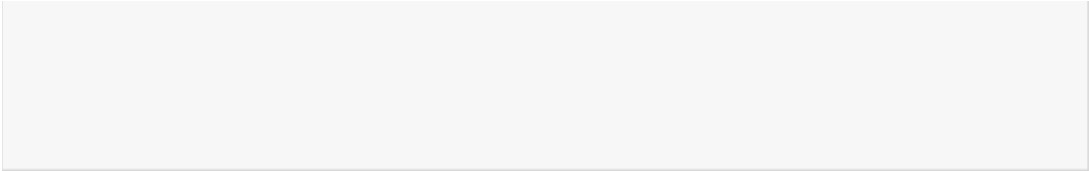
model.compile( loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy']

)

# 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 -

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 -

val\_loss: 1.5237 - val\_accuracy: 0.5673 Epoch 16/25

98/98 [==============================] - 596s 6s/step - loss: 0.0437 - accuracy: 0.9888 -

val\_loss: 1.4307 - val\_accuracy: 0.6140 Epoch 17/25

98/98 [==============================] - 602s 6s/step - loss: 0.0428 - accuracy: 0.9877 -

val\_loss: 1.2403 - val\_accuracy: 0.6433 Epoch 18/25

98/98 [==============================] - 605s 6s/step - loss: 0.0359 - accuracy: 0.9949 -

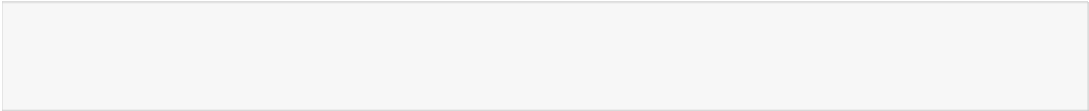
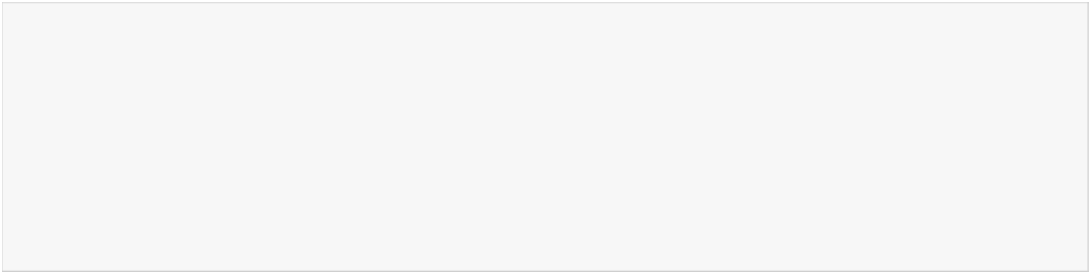
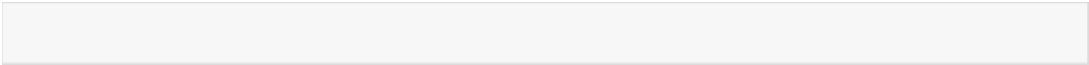
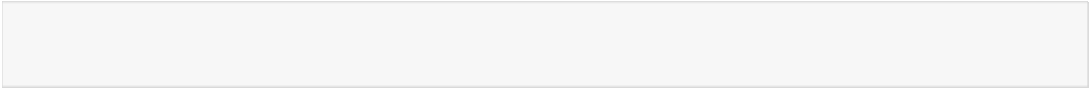
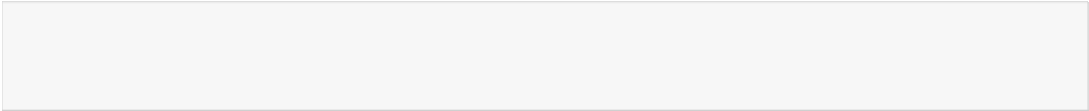
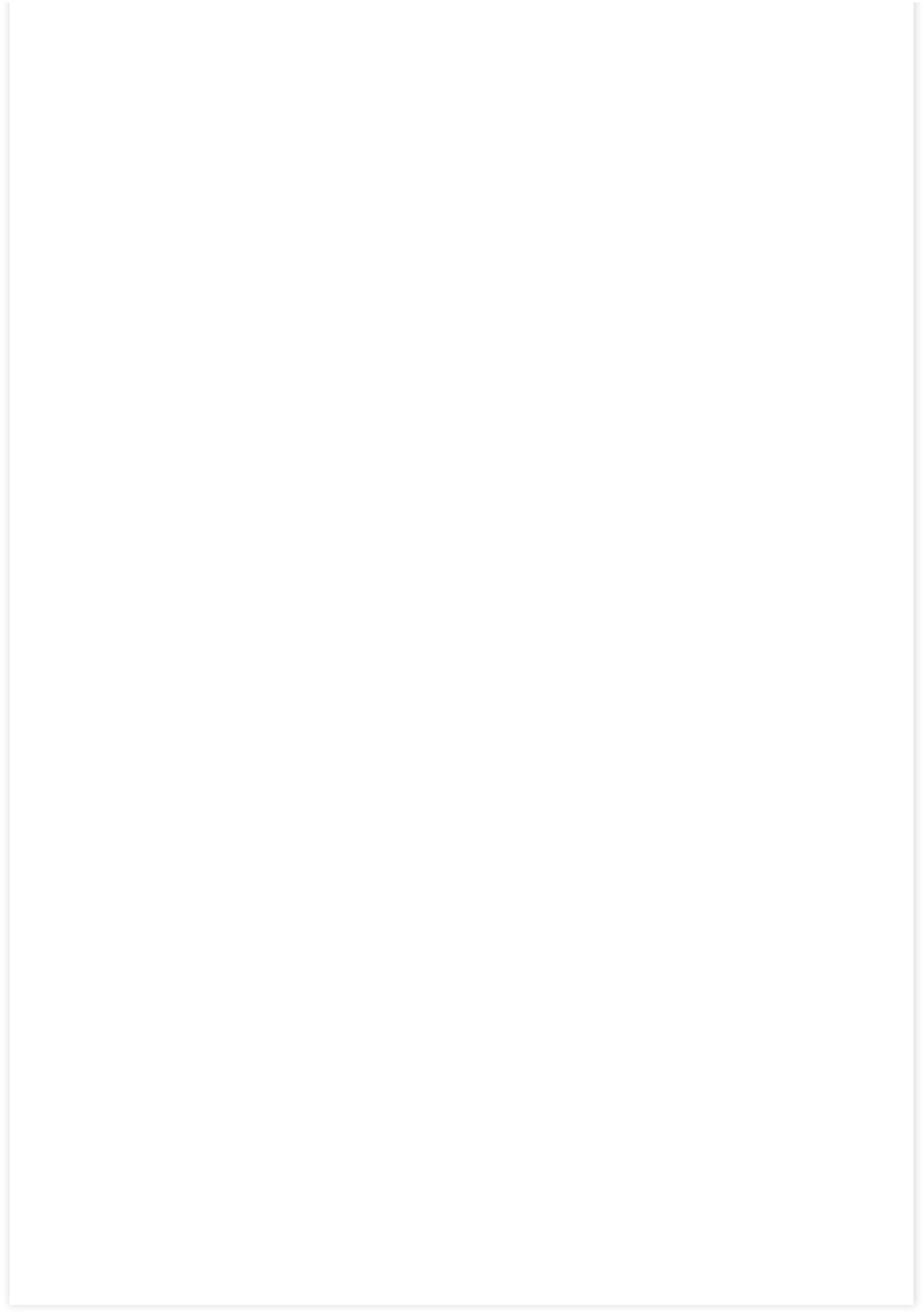
val\_loss: 1.3156 - val\_accuracy: 0.6433 Epoch 19/25

98/98 [==============================] - 598s 6s/step - loss: 0.0289 - accuracy: 0.9959 -

val\_loss: 1.4142 - val\_accuracy: 0.6140 Epoch 20/25

98/98 [==============================] - 594s 6s/step - loss: 0.0256 - accuracy: 0.9980 -

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| val\_loss: 1.3567 - val\_accuracy: 0.6316 |  | | | | | | |
| Epoch 21/25 |
| 98/98 [==============================] - | 598s | 6s/step | - loss: | 0.0248 | - accuracy: | 0.9990 | - |
| val\_loss: 1.3492 - val\_accuracy: 0.6257 |  |  |  |  |  |  |  |
| Epoch 22/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 596s | 6s/step | - loss: | 0.0222 | - accuracy: | 1.0000 | - |
| val\_loss: 1.3326 - val\_accuracy: 0.6491 |  |  |  |  |  |  |  |
| Epoch 23/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 597s | 6s/step | - loss: | 0.0137 | - accuracy: | 0.9990 | - |
| val\_loss: 1.4157 - val\_accuracy: 0.6199 |  |  |  |  |  |  |  |
| Epoch 24/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 595s | 6s/step | - loss: | 0.0398 | - accuracy: | 0.9888 | - |
| val\_loss: 1.4562 - val\_accuracy: 0.6257 |  |  |  |  |  |  |  |
| Epoch 25/25 |  |  |  |  |  |  |  |
| 98/98 [==============================] - | 597s | 6s/step | - loss: | 0.0292 | - accuracy: | 0.9939 | - |
| val\_loss: 1.5857 - val\_accuracy: 0.5965  **8. Save The Model** |  |  |  |  |  |  |  |



In [28]:

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 [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))

1/1 [==============================] - 1s 674ms/step

minor