

**TEAM ID -  
PNT2022TMID46426**

## ▾ Import and unzip the dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
#unzip the downloaded dataset
!unzip '/content/drive/MyDrive/damage_vehicle.zip'
```

Archive:

- /content/drive/MyDrive/damage\_vehicle.zip
- creating: damage\_vehicle/
- creating: damage\_vehicle/body/
- creating: damage\_vehicle/body/training/
- creating: damage\_vehicle/body/training/00-front/
- inflating: damage\_vehicle/body/training/00-front/0001.jpeg
- inflating: damage\_vehicle/body/training/00-front/0002.JPEG
- inflating: damage\_vehicle/body/training/00-front/0003.JPEG
- inflating: damage\_vehicle/body/training/00-front/0004.JPEG
- inflating: damage\_vehicle/body/training/00-front/0005.JPEG
- inflating: damage\_vehicle/body/training/00-front/0006.JPEG
- inflating: damage\_vehicle/body/training/00-front/0007.JPEG
- inflating: damage\_vehicle/body/training/00-front/0008.jpeg
- inflating: damage\_vehicle/body/training/00-front/0009.JPEG
- inflating: damage\_vehicle/body/training/00-front/0010.JPEG
- inflating: damage\_vehicle/body/training/00-front/0011.JPEG
- inflating: damage\_vehicle/body/training/00-front/0012.jpeg
- inflating: damage\_vehicle/body/training/00-front/0013.JPEG
- inflating: damage\_vehicle/body/training/00-front/0014.JPEG
- inflating: damage\_vehicle/body/training/00-front/0015.JPEG
- inflating: damage\_vehicle/body/training/00-front/0016.JPEG
- inflating: damage\_vehicle/body/training/00-front/0017.JPEG
- inflating: damage\_vehicle/body/training/00-front/0018.JPEG
- inflating: damage\_vehicle/body/training/00-front/0019.JPEG
- inflating: damage\_vehicle/body/training/00-front/0020.jpeg
- inflating: damage\_vehicle/body/training/00-front/0021.JPEG
- inflating: damage\_vehicle/body/training/00-front/0022.JPEG
- inflating: damage\_vehicle/body/training/00-front/0023.JPEG
- inflating: damage\_vehicle/body/training/00-front/0024.JPEG
- inflating: damage\_vehicle/body/training/00-front/0025.jpeg
- inflating: damage\_vehicle/body/training/00-front/0026.JPEG
- inflating: damage\_vehicle/body/training/00-front/0027.JPEG
- inflating: damage\_vehicle/body/training/00-front/0028.JPEG
- inflating: damage\_vehicle/body/training/00-front/0029.JPEG
- inflating: damage\_vehicle/body/training/00-front/0030.JPEG
- inflating: damage\_vehicle/body/training/00-front/0031.JPEG

vehicle/body/training/00-front/0032.JPEGinflating:  
damage vehicle/body/training/00-front/0033.JPEG  
inflating: damage vehicle/body/training/00-  
front/0034.JPEGinflating: damage  
vehicle/body/training/00-front/0035.jpeginflating:  
damage vehicle/body/training/00-front/0036.JPEG  
inflating: damage vehicle/body/training/00-  
front/0037.JPEGinflating: damage  
vehicle/body/training/00-front/0038.JPEGinflating:  
damage vehicle/body/training/00-front/0039.JPEG  
inflating: damage vehicle/body/training/00-  
front/0040.JPEG inflating: damage  
vehicle/body/training/00-front/0041.JPEGinflating:  
damage vehicle/body/training/00-front/0042.JPEG  
inflating: damage vehicle/body/training/00-  
front/0043.JPEGinflating: damage  
vehicle/body/training/00-front/0044.JPEGinflating:  
damage vehicle/body/training/00-front/0045.JPEG  
inflating: damage vehicle/body/training/00-  
front/0046.jpeginflating: damage  
vehicle/body/training/00-front/0047.JPEGinflating:  
damage vehicle/body/training/00-front/0048.JPEG  
inflating: damage vehicle/body/training/00-  
front/0049.JPEGinflating: damage  
vehicle/body/training/00-front/0050.JPEGinflating:  
damage vehicle/body/training/00-front/0051.JPEG  
inflating: damage vehicle/body/training/00-  
front/0052.JPEGinflating: damage  
vehicle/body/training/00-front/0053.JPEG

## ▼ Image Preprocessing

### 1. Import The ImageDataGenerator Library

```
# Import required lib
```

B

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

## 2. Configure ImageDataGenerator Class

```
#Creating augmentation on training variable
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range = 0.1,
                                   zoom_range=0.1,
                                   horizontal_flip=True)

# Creating augmentation on testing variable

test_datagen = ImageDataGenerator(rescale=1./255)
```

## 3. Apply ImageDataGenerator Functionality To Trainset And Testset

```
# Passing training data to train variable for body

xtrain = train_datagen.flow_from_directory('/content/damage_vehicle/body/training',
                                           target_size=(224,224),
                                           class_mode='categorical',
                                           batch_size=10)
```

Found 979 images belonging to 3 classes.

```
# Passing testing data to test variable for body

xtest = test_datagen.flow_from_directory('/content/damage_vehicle/body/validation',
                                         target_size=(224,224),
                                         class_mode='categorical',
                                         batch_size=10)
```

Found 171 images belonging to 3 classes.

```
# Passing training data to train variable for level

x_train = train_datagen.flow_from_directory('/content/damage_vehicle/level/training',
                                           target_size=(224,224),
                                           class_mode='categorical',
                                           batch_size=10)
```

Found 979 images belonging to 3 classes.

```
# Passing testing data to test variable for level

x_test = test_datagen.flow_from_directory('/content/damage_vehicle/level/validation',
                                         target_size=(224,224),
                                         class_mode='categorical',
                                         batch_size=10)
```

Found 171 images belonging to 3 classes.

**M**

**o**

**d**

**el**

**B**

**ui**

**ld**

**in**

**g**

**F**

**o**

**r**

**B**

**o**

**d**

**y**

## **1. Importing The Model Building Libraries**

```
#Import the library
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from glob import glob
```

1

```
import numpy as np

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/damage_vehicle/body/training'
valid_path = '/content/damage_vehicle/body/validation'
```

```
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/58889256/58889256](https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels/58889256/58889256) [=====] - 3s 0us/step

## 3. Adding Flatten Layer

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

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

```
folders
```

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

```
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_pool1 (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(
    xtrain,
    validation_data=xtest,
    epochs=25,
    steps_per_epoch=len(xtrain),
    validation_steps=len(xtest)
)
```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:6: UserWarning: `Model.fit\_generator` is deprecated and will be

```
Epoch 1/25
98/98 [=====] - 23 146ms/ste - loss: 1.207 - accuracy: 0.546 - val_loss: 1.290 - val_accuracy:
Epoch ==] s p 7 5 0
2/25
98/98 [=====] 13 128ms/ste - - loss: 0.836 - - 0.702 - - 0.866 - -
Epoc ==] 3/25 s p 4 accuracy: 8 val_loss: 5 val_accuracy:
h [=====] - - loss:
98/98 ==] 13 128ms/ste 0.529 - - 0.799 - - 1.326 - -
s p 3 accuracy: 8 val_loss: 0 val_accuracy:
Epoch 4/25
98/98 [=====] - 12 127ms/ste - loss: 0.397 - accuracy: 0.861 - val_loss: 0.984 - val_accuracy:
==] s p 8 1 2
```



```

Epoch 5/25
98/98 [=====] 12 127ms/ste - - loss: 0.278 - - 0.903 - - 0.939 - -
Epoch 6/25
98/98 [=====] 13 128ms/ste - - loss: 0.269 - - 0.907 - - 0.989 - -
Epoch 7/25
98/98 [=====] - 12 127ms/ste - loss: 0.178 - accuracy: 0.944 - val_loss: 1.005 - val_accuracy:
Epoch 8/25
98/98 [=====] - 13 129ms/ste - loss: 0.167 - accuracy: 0.946 - val_loss: 1.169 - val_accuracy:
Epoch 9/25
98/98 [=====] 13 129ms/ste - - loss: 0.127 - - 0.956 - - 1.005 - -
Epoch 10/25
98/98 [=====] 13 128ms/ste - - loss: 0.118 - - 0.959 - - 1.062 - -
Epoch 11/25
98/98 [=====] - 13 130ms/ste - loss: 0.096 - accuracy: 0.974 - val_loss: 1.121 - val_accuracy:
Epoch 12/25

```

```

98/98 [===== - 129ms/ste - loss: 0.085 - accuracy: 0.976 - val_loss: 1.028 - val_accuracy:
Epoch =] 13 p 7 5 4
13 s
98/98 [===== - 129ms/ste - - loss: 0.058 - - 0.983 - - val_loss: 1.115 - -
Epoch =] 14/25 13 p 2 7 3 val_accuracy:
h [===== s - - loss:
98/98 =] 129ms/ste 0.068 - - 0.987 - - val_loss: 1.103 - -
- p 8 accuracy: 7 3 val_accuracy:
13 s
Epoch 15/25
98/98 [===== - 131ms/ste - loss: 0.070 - accuracy: 0.986 - val_loss: 1.073 - val_accuracy:
=] 13 p 9 7 0
13 s
Epoch 16/25
98/98 [===== - 128ms/ste - loss: 0.089 - accuracy: 0.977 - val_loss: 1.122 - val_accuracy:
=] 13 p 5 5 5
13 s
Epoch 17/25
98/98 [===== - 129ms/ste - - loss: 0.060 - - 0.991 - - val_loss: 1.293 - -
Epoch =] 18/25 13 p 9 8 7 val_accuracy:
98/98 [===== s - - loss:
=] 128ms/ste 0.099 - - 0.971 - - val_loss: 1.175 - -
- p 8 accuracy: 4 4 val_accuracy:
13 s
Epoch 19/25
98/98 [===== - 128ms/ste - loss: 0.072 - accuracy: 0.984 - val_loss: 1.507 - val_accuracy:
=] 13 p 8 7 4
13 s
Epoch 20/25
98/98 [===== - 129ms/ste - - loss: 0.097 - - 0.971 - - val_loss: 1.468 - -
Epoch =] 21/25 13 p 2 4 4 val_accuracy:
98/98 [===== s - - loss:
=] 131ms/ste 0.040 - - 0.990 - - val_loss: 1.421 - -
- p 4 accuracy: 8 5 val_accuracy:
13 s
Epoch 22/25
98/98 [===== - 131ms/ste - loss: 0.085 - accuracy: 0.986 - val_loss: 1.477 - val_accuracy:
=] 13 p 4 7 2
13 s
Epoch 23/25
98/98 [===== - 128ms/ste - loss: 0.039 - accuracy: 0.991 - val_loss: 1.430 - val_accuracy:
=] 13 p 9 8 6
13 s
Epoch 24/25
98/98 [===== - 129ms/ste - - loss: 0.040 - - 0.990 - - val_loss: 1.456 - -
Epoch =] 13 p 0 8 2 val_accuracy:
h 25/25 s - - loss:
98/98 [===== 129ms/ste 0.169 - - 0.938 - - val_loss: 1.680 - -
=] - p 2 accuracy: 7 5 val_accuracy:
13 s

```

## 8. Save The Model

```

from tensorflow.keras.models import load_model

model.save('/content/damage_vehicle/Model/body.h5')

```

## 9. Test The Model

```

from tensorflow.keras.models
import load_modelimport cv2
from skimage.transform import resize

```

```
model = load_model('/content/damage_vehicle/Model/body.h5')
```

```
def detect(frame):
```

```
    img = cv2.resize(frame,(224,224))
```

```
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
```

```
    i
```

```
    f
```

```
    (
```

```
    n
```

```
    p
```

```
    .
```

```
    m
```

```
    a
```

```
    x
```

```
    (
```

```
    i
```

```
    m
```

```
    g
```

```
    )
```

```
    >
```

```
    1
```

```
    )
```

```
    :
```

```
    i
```

```
    m
```

```
    g
```

```
    =
```

```
    i
```

```
    m
```

```
    g
```

```
    /
```

```
    2
```

```
    5
```

```
    5
```

```
    .
```

```
    0
```

```
    img =
```

```
    np.array([img])
```

```
    prediction =
```

```
    model.predict(i
```

```
    mg)label =
```

```
    ["front", "rear", "
```

```
    side"]
```

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

```
    return preds
```

```
import numpy as np
```

```
data = "/content/damage_vehicle/body/training/00 -
```

```
front/0002.JPEG"image = cv2.imread(data)
```

```
print(detect(image))
```

```
1/1 [=====] - 0s 148ms/step  
front
```

**M**

**o**

▼

**d**

**el**

**B**

**ui**

**ld**

**in**

**g**

**F**

**o**

**r**

**L**

**e**

**v**

**el**

## **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/damage_vehicle/level/training'
valid_path = '/content/damage_vehicle/level/validation'

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

## 3. Adding Flatten Layer

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

folders = glob('/content/damage_vehicle/level/training/*')
```

```
folders
```

```
['/content/damage_vehicle/level/training/03-
severe', '/content/damage
vehicle/level/training/02-moderate',
'/content/damage_vehicle/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_1"
```

Layer (type)	Output Shape	Param #
input_2 (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_1 (Flatten)	(None, 25088)	0
dense_1 (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(
    x_train,
    validation_data=x_test,
    epochs=25,
    steps_per_epoch=len(x_train),
    validation_steps=len(x_test)
)

```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:6: UserWarning: `Model.fit\_generator` is deprecated and will be

```

Epoch 1/25
98/98 [=====] 14 133ms/ste - - loss: 1.162 - - 0.549 - - 1.155 - -
Epoch ==] 2/25 s p 9 accuracy: 5 val_loss: 9 val_accuracy:
h [=====] - - loss:
98/98 ==] 13 130ms/ste 0.715 - - 0.708 - - 0.964 - -
s p 7 accuracy: 9 val_loss: 3 val_accuracy:

Epoch 3/25
98/98 [=====] - 13 130ms/ste - loss: 0.497 - accuracy: 0.816 - val_loss: 1.566 - val_accuracy:
==] s p 8 1 3

Epoch 4/25
98/98 [=====] - 13 128ms/ste - loss: 0.527 - accuracy: 0.786 - val_loss: 1.600 - val_accuracy:
==] s p 7 5 3

Epoch 5/25
98/98 [=====] 13 128ms/ste - - loss: 0.376 - - 0.846 - - 1.192 - -
Epoch ==] 6/25 s p 3 accuracy: 8 val_loss: 5 val_accuracy:
98/98 - - loss:

```

```

[===== 13 128ms/ste 0.244 - - 0.920 - - 1.035 - -
==] s p 5 accuracy: 3 val_loss: 4 val_accuracy:

Epoch 7/25
98/98 [===== - 13 128ms/ste - loss: 0.190 - - accuracy: 0.934 - - val_loss: 1.215 - - val_accuracy:
==] s p 2 6 5

Epoch 8/25
98/98 [===== - 13 128ms/ste - loss: 0.132 - - accuracy: 0.957 - - val_loss: 1.090 - - val_accuracy:
==] s p 7 1 2

Epoch 9/25
98/98 [===== 13 127ms/ste - - loss: 0.120 - - 0.954 - - 1.128 - -
==] s p 6 accuracy: 0 val_loss: 2 val_accuracy:

Epoch 10/25
98/98 [===== 13 128ms/ste - - loss: 0.118 - - 0.959 - - 1.131 - -
==] s p 1 accuracy: 1 val_loss: 1 val_accuracy:

Epoch 11/25
98/98 [===== - 13 128ms/ste - loss: 0.091 - - accuracy: 0.976 - - val_loss: 1.153 - - val_accuracy:
==] s p 0 5 8

Epoch 12/25
98/98 [===== 12 127ms/ste - - loss: 0.081 - - 0.980 - - 1.220 - -
==] s p 3 accuracy: 6 val_loss: 9 val_accuracy:

Epoch 13/25
98/98 [===== 13 128ms/ste - - loss: 0.060 - - 0.985 - - 1.254 - -
==] s p 3 accuracy: 7 val_loss: 5 val_accuracy:

Epoch 14/25

```



98/98	[=====	- 12	127ms/ste	- loss:	0.047	- accuracy:	0.994	- val_loss:	1.160	- val_accuracy:
Epoch ==]		s	p		4		9		9	
15/25										
98/98	[=====	13	129ms/ste	- - loss:	0.036	- -	0.995	- -	1.168	- -
Epoc ==] 16/25		s	p		6	accuracy:	9	val_loss:	8	val_accuracy:
h	[=====			- - loss:						
98/98 ==]		13	128ms/ste		0.049	- -	0.988	- -	1.185	- -
		s	p		3	accuracy:	8	val_loss:	0	val_accuracy:
Epoch 17/25										
98/98	[=====	- 13	128ms/ste	- loss:	0.032	- accuracy:	0.993	- val_loss:	1.188	- val_accuracy:
==]		s	p		0		9		4	
Epoch 18/25										
98/98	[=====	- 13	129ms/ste	- loss:	0.036	- accuracy:	0.993	- val_loss:	1.289	- val_accuracy:
==]		s	p		3		9		7	
Epoch 19/25										
98/98	[=====	13	128ms/ste	- - loss:	0.029	- -	0.994	- -	1.249	- -
Epoch ==] 20/25		s	p		8	accuracy:	9	val_loss:	9	val_accuracy:
98/98	[=====			- - loss:						
==]		13	130ms/ste		0.025	- -	0.998	- -	1.280	- -
		s	p		0	accuracy:	0	val_loss:	1	val_accuracy:
Epoch 21/25										
98/98	[=====	- 13	129ms/ste	- loss:	0.032	- accuracy:	0.995	- val_loss:	1.236	- val_accuracy:
==]		s	p		9		9		6	
Epoch 22/25										
98/98	[=====	13	128ms/ste	- - loss:	0.017	- -	1.000	- -	1.290	- -
Epoch ==] 23/25		s	p		0	accuracy:	0	val_loss:	1	val_accuracy:
98/98	[=====			- - loss:						
==]		13	130ms/ste		0.021	- -	1.000	- -	1.269	- -
		s	p		6	accuracy:	0	val_loss:	7	val_accuracy:
Epoch 24/25										
98/98	[=====	- 13	128ms/ste	- loss:	0.036	- accuracy:	0.990	- val_loss:	1.421	- val_accuracy:
==]		s	p		5		8		4	
Epoc 25/25										
h	[=====	- 13	129ms/ste	- loss:	0.038	- accuracy:	0.993	- val_loss:	1.421	- val_accuracy:
98/98 ==]		s	p		0		9		9	

<

>

## 8. Save The Model

```
from tensorflow.keras.models import load_model

model.save('/content/damage_vehicle/Model/level.h5')
```

## 9. Test The Model

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

```
model = load_model('/content/damage_vehicle/Model/level.h5')
```

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

```
i
f
(
n
p
.
m
a
x
(
i
m
g
```

```

)
>
1
)
:
i
m
g
=
i
m
g
/
2
5
5
.
0
img =
np.array([img])
prediction =
model.predict(img
)
label = ["minor", "moderate", "severe"]
preds =
label[np.argmax(predicti
on)]return preds

```

```
import numpy as np
```

```

data = "/content/damage_vehicle/level/validation/01 -
minor/0005.JPEG"image = cv2.imread(data)
print(detect(image))

```

```

1/1 [=====] - 0s 142ms/step
minor

```

