Team id : PNT2022TMID05405

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



# Image Preprocessing

## 1. Import The ImageDataGenerator Library



# Import required lib

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.

**Model Building**

# For Body

## 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 |

import numpy as np

import matplotlib.pyplot as plt

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\_ke](https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5) 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\_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( 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 [==============================] - 23s 146ms/step - loss: 1.2077 - accuracy: 0.5465 - val\_loss: 1.2900 - val\_accuracy:

Epoch 2/25

98/98 [==============================] - 13s 128ms/step - loss: 0.8364 - accuracy: 0.7028 - val\_loss: 0.8665 - val\_accuracy:

Epoch 3/25

98/98 [==============================] - 13s 128ms/step - loss: 0.5293 - accuracy: 0.7998 - val\_loss: 1.3260 - val\_accuracy:

Epoch 4/25

98/98 [==============================] - 12s 127ms/step - loss: 0.3978 - accuracy: 0.8611 - val\_loss: 0.9842 - val\_accuracy:

Epoch 5/25

98/98 [==============================] - 12s 127ms/step - loss: 0.2783 - accuracy: 0.9030 - val\_loss: 0.9397 - val\_accuracy:

Epoch 6/25

98/98 [==============================] - 13s 128ms/step - loss: 0.2690 - accuracy: 0.9070 - val\_loss: 0.9892 - val\_accuracy:

Epoch 7/25

98/98 [==============================] - 12s 127ms/step - loss: 0.1788 - accuracy: 0.9448 - val\_loss: 1.0052 - val\_accuracy:

Epoch 8/25

98/98 [==============================] - 13s 129ms/step - loss: 0.1671 - accuracy: 0.9469 - val\_loss: 1.1693 - val\_accuracy:

Epoch 9/25

98/98 [==============================] - 13s 129ms/step - loss: 0.1277 - accuracy: 0.9561 - val\_loss: 1.0058 - val\_accuracy:

Epoch 10/25

98/98 [==============================] - 13s 128ms/step - loss: 0.1184 - accuracy: 0.9591 - val\_loss: 1.0620 - val\_accuracy:

Epoch 11/25

98/98 [==============================] - 13s 130ms/step - loss: 0.0963 - accuracy: 0.9745 - val\_loss: 1.1219 - val\_accuracy:

Epoch 12/25



98/98 [==============================] - 13s 129ms/step - loss: 0.0857 - accuracy: 0.9765 - val\_loss: 1.0284 - val\_accuracy:

Epoch 13/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0582 - accuracy: 0.9837 - val\_loss: 1.1153 - val\_accuracy:

Epoch 14/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0688 - accuracy: 0.9877 - val\_loss: 1.1033 - val\_accuracy:

Epoch 15/25

98/98 [==============================] - 13s 131ms/step - loss: 0.0709 - accuracy: 0.9867 - val\_loss: 1.0730 - val\_accuracy:

Epoch 16/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0895 - accuracy: 0.9775 - val\_loss: 1.1225 - val\_accuracy:

Epoch 17/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0609 - accuracy: 0.9918 - val\_loss: 1.2937 - val\_accuracy:

Epoch 18/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0998 - accuracy: 0.9714 - val\_loss: 1.1754 - val\_accuracy:

Epoch 19/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0728 - accuracy: 0.9847 - val\_loss: 1.5074 - val\_accuracy:

Epoch 20/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0972 - accuracy: 0.9714 - val\_loss: 1.4684 - val\_accuracy:

Epoch 21/25

98/98 [==============================] - 13s 131ms/step - loss: 0.0404 - accuracy: 0.9908 - val\_loss: 1.4215 - val\_accuracy:

Epoch 22/25

98/98 [==============================] - 13s 131ms/step - loss: 0.0854 - accuracy: 0.9867 - val\_loss: 1.4772 - val\_accuracy:

Epoch 23/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0399 - accuracy: 0.9918 - val\_loss: 1.4306 - val\_accuracy:

Epoch 24/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0400 - accuracy: 0.9908 - val\_loss: 1.4562 - val\_accuracy:

Epoch 25/25

98/98 [==============================] - 13s 129ms/step - loss: 0.1692 - accuracy: 0.9387 - val\_loss: 1.6805 - val\_accuracy:



## 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\_model import 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)    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 |

import numpy as np

data = "/content/damage vehicle/body/training/00-front/0002.JPEG" image = cv2.imread(data) print(detect(image))

1/[[1]](#footnote-1) [==============================] - 0s 148ms/step front

**Model Building**

# For Level

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 [==============================] - 14s 133ms/step - loss: 1.1629 - accuracy: 0.5495 - val\_loss: 1.1559 - val\_accuracy:

Epoch 2/25

98/98 [==============================] - 13s 130ms/step - loss: 0.7157 - accuracy: 0.7089 - val\_loss: 0.9643 - val\_accuracy:

Epoch 3/25

98/98 [==============================] - 13s 130ms/step - loss: 0.4978 - accuracy: 0.8161 - val\_loss: 1.5663 - val\_accuracy:

Epoch 4/25

98/98 [==============================] - 13s 128ms/step - loss: 0.5277 - accuracy: 0.7865 - val\_loss: 1.6003 - val\_accuracy:

Epoch 5/25

98/98 [==============================] - 13s 128ms/step - loss: 0.3763 - accuracy: 0.8468 - val\_loss: 1.1925 - val\_accuracy:

Epoch 6/25

98/98 [==============================] - 13s 128ms/step - loss: 0.2445 - accuracy: 0.9203 - val\_loss: 1.0354 - val\_accuracy:

Epoch 7/25

98/98 [==============================] - 13s 128ms/step - loss: 0.1902 - accuracy: 0.9346 - val\_loss: 1.2155 - val\_accuracy:

Epoch 8/25

98/98 [==============================] - 13s 128ms/step - loss: 0.1327 - accuracy: 0.9571 - val\_loss: 1.0902 - val\_accuracy:

Epoch 9/25

98/98 [==============================] - 13s 127ms/step - loss: 0.1206 - accuracy: 0.9540 - val\_loss: 1.1282 - val\_accuracy:

Epoch 10/25

98/98 [==============================] - 13s 128ms/step - loss: 0.1181 - accuracy: 0.9591 - val\_loss: 1.1311 - val\_accuracy:

Epoch 11/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0910 - accuracy: 0.9765 - val\_loss: 1.1538 - val\_accuracy:

Epoch 12/25

98/98 [==============================] - 12s 127ms/step - loss: 0.0813 - accuracy: 0.9806 - val\_loss: 1.2209 - val\_accuracy:

Epoch 13/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0603 - accuracy: 0.9857 - val\_loss: 1.2545 - val\_accuracy:

Epoch 14/25



98/98 [==============================] - 12s 127ms/step - loss: 0.0474 - accuracy: 0.9949 - val\_loss: 1.1609 - val\_accuracy:

Epoch 15/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0366 - accuracy: 0.9959 - val\_loss: 1.1688 - val\_accuracy:

Epoch 16/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0493 - accuracy: 0.9888 - val\_loss: 1.1850 - val\_accuracy:

Epoch 17/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0320 - accuracy: 0.9939 - val\_loss: 1.1884 - val\_accuracy:

Epoch 18/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0363 - accuracy: 0.9939 - val\_loss: 1.2897 - val\_accuracy:

Epoch 19/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0298 - accuracy: 0.9949 - val\_loss: 1.2499 - val\_accuracy:

Epoch 20/25

98/98 [==============================] - 13s 130ms/step - loss: 0.0250 - accuracy: 0.9980 - val\_loss: 1.2801 - val\_accuracy:

Epoch 21/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0329 - accuracy: 0.9959 - val\_loss: 1.2366 - val\_accuracy:

Epoch 22/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0170 - accuracy: 1.0000 - val\_loss: 1.2901 - val\_accuracy:

Epoch 23/25

98/98 [==============================] - 13s 130ms/step - loss: 0.0216 - accuracy: 1.0000 - val\_loss: 1.2697 - val\_accuracy:

Epoch 24/25

98/98 [==============================] - 13s 128ms/step - loss: 0.0365 - accuracy: 0.9908 - val\_loss: 1.4214 - val\_accuracy:

Epoch 25/25

98/98 [==============================] - 13s 129ms/step - loss: 0.0380 - accuracy: 0.9939 - val\_loss: 1.4219 - val\_accuracy:



## 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\_model import 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)    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 |

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



[Colab paid products](https://colab.research.google.com/signup?utm_source=footer&utm_medium=link&utm_campaign=footer_links) - [Cancel contracts here](https://colab.research.google.com/cancel-subscription)





1. 1. Importing The Model Building Libraries  [↑](#footnote-ref-1)