## Import and Unzip the Dataset

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.

# unzip the the downloaded dateset

!unzip '/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body-20221104T08

inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/bod

# unzip the the downloaded dateset

!unzip '/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level-20221104T0

inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/le inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev inflating: /content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/lev

## Image Preprocessing

1. Import The ImageDataGenerator Library

# import required library

from tensorflow.keras.preprocessing.image import ImageDataGenerator

## Configure ImageDataGeneratorLoss

#creating augmentaiton on training variable on train\_datagen 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)

## Apply ImageDataGenerator Functionality to Trainset and Testset

#Passing training data to train variable for body

xtrain=train\_datagen.flow\_from\_directory('/content/drive/MyDrive/workSpace/GouseRabbani3

Found 2300 images belonging to 3 classes.

#Passing testing data to train variable for body

xtest=test\_datagen.flow\_from\_directory('/content/drive/MyDrive/workSpace/GouseRabbani330

Found 2300 images belonging to 3 classes.

#Passing training data to train variable for level

xtrain=train\_datagen.flow\_from\_directory('/content/drive/MyDrive/workSpace/GouseRabbani3 Found 2300 images belonging to 3 classes.

#Passing training data to test variable for level

xtest=test\_datagen.flow\_from\_directory('/content/drive/MyDrive/workSpace/GouseRabbani330 Found 2300 images belonging to 3 classes.

# Model Building

## Import the model building libraries

#import the libraries

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Flatten, Input from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.vgg16 import VGG16, preprocess\_input from glob import glob

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

## Loading the model

IMAGE\_SIZE = (224, 224)

train\_path = '/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body/train test\_path = '/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body/valida

vgg16 = VGG16(input\_shape=(IMAGE\_SIZE[0], IMAGE\_SIZE[1], 3), weights='imagenet', include

Downloading data from [https://storage.googleapis.com/tensorflow/keras-applications/v](https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5) 58889256/58889256 [==============================] - 0s 0us/step

## Add Flattening Layer

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

folders=glob('/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body/train

folders = [

"/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body/training/00-fr "/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body/training/01-re "/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body/training/02-si

]

x = Flatten()(vgg16.output)

len(folders) 3

## Adding Output layer

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

## Creating a Model Object

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

model.summary()

Model: "model\_1"

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\_2 (Flatten) | | (None, | 25088) | | | 0 |
| dense\_1 (Dense) | | (None, | 3) | | | 75267 |

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

Total params: 14789955 (56.42 MB)

Trainable params: 75267 (294.01 KB)

Non-trainable params: 14714688 (56.13 MB)

## Configuring the Learning Process

model.compile(

loss='categorical\_crossentropy', optimizer='adam',

metrics=['accuracy']

)

## Train the Model

import sys

g = model.fit\_generator( xtrain,

validation\_data=xtest, epochs=25,

steps\_per\_epoch=979//10, validation\_steps=171//10

)

<ipython-input-71-208b9b864d48>:3: UserWarning: `Model.fit\_generator` is deprecated g = model.fit\_generator(

Epoch 1/25

27/97 [=======>......................] - ETA: 24:07 - loss: 1.1589 - accuracy: 0.447

## Save the Model

from tensorflow.keras.models import load\_model model.save('body.h5')

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarni saving\_api.save\_model(

## Test the Model

def defect(frame):

# Convert the frame to a NumPy array img = np.array(frame)

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

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) if (np.max(img) > 1):

img = img / 255

img = np.array([img])

prediction = model.predict(img)

label = ['front', 'rear', 'side']

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

import cv2

# Load the JPEG image

image = cv2.imread('/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body

# Use the defect function to predict the defect prediction = defect(image)

# Print the prediction print(prediction)

1/1 [==============================] - 1s 719ms/step

rear

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

from skimage.transform import resize model = load\_model('body.h5')

def defect(frame):

import numpy as np

image = np.array(frame)

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

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) if (np.max(img)>1):

img = img/255

img = np.array([img])

prediction = model.predict(img) label = ['front','rear','side']

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

image = cv2.imread('/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/body prediction = defect (image)

print(prediction)

1/1 [==============================] - 1s 694ms/step

rear

Start coding or ge nerate

with AI.

# Model Building

**For Level**

## Importing the model building libraries

#import the libraries

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Flatten, Input from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.vgg16 import VGG16, preprocess\_input from glob import glob

## Loading the Model

IMAGE\_SIZE = (224, 224)

train\_path = '/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level/trai test\_path = '/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level/valid

vgg16 = VGG16(input\_shape=(IMAGE\_SIZE[0], IMAGE\_SIZE[1], 3), weights='imagenet', include

## Adding Flattening Layer

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

folders=glob('/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level/trai

folders = [

"/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level/level/trainin "/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level/level/trainin "/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level/level/trainin

]

x = Flatten()(vgg16.output)

len(folders) 3

## Adding Output Layer

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

## Creating the Output Model

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

model.summary()

Model: "model\_2"

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\_3 (Flatten) | (None, | 25088) | | 0 |
| dense\_2 (Dense) | (None, | 3) | | 75267 |

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

Total params: 14789955 (56.42 MB)

Trainable params: 75267 (294.01 KB)

Non-trainable params: 14714688 (56.13 MB)

## Configure the Learning Process

model.compile(

loss='categorical\_crossentropy', optimizer='adam',

metrics=['accuracy']

)

## Train the Model

import sys

g = model.fit\_generator( xtrain,

validation\_data=xtest, epochs=25,

steps\_per\_epoch=979//10, validation\_steps=171//10

)

<ipython-input-107-208b9b864d48>:3: UserWarning: `Model.fit\_generator` is deprecated g = model.fit\_generator(

Epoch 1/25

72/97 [=====================>........] - ETA: 8:31 - loss: 1.3987 - accuracy: 0.4183

97/97 [==============================] - 1811s 19s/step - loss: 1.3987 - accuracy: 0

## Save the Model

from tensorflow.keras.models import load\_model model.save('level.h5')

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarni saving\_api.save\_model(

## Test the Model

def defect(frame):

# Convert the frame to a NumPy array img = np.array(frame)

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

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

import cv2

if (np.max(img) > 1):

img = img / 255

img = np.array([img])

prediction = model.predict(img)

label = ['front', 'rear', 'side']

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

# Load the JPEG image

image = cv2.imread('/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/leve

# Use the defect function to predict the defect prediction = defect(image)

# Print the prediction print(prediction)

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

front

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

from skimage.transform import resize model = load\_model('body.h5')

def defect(frame):

import numpy as np

image = np.array(frame)

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

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) if (np.max(img)>1):

img = img/255

img = np.array([img])

prediction = model.predict(img) label = ['front','rear','side']

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

image = cv2.imread('/content/drive/MyDrive/workSpace/GouseRabbani330/DATACOLLECTION/level/ prediction = defect (image)

print(prediction)

1/1 [==============================] - 1s 826ms/step

rear

Start coding or ge nerate

with AI.