**Project Development Phase**

**Sprint-1**

**Model Building (Level)**

|  |  |
| --- | --- |
| Date | 08 November 2022 |
| Team ID | PNT2022TMID36535 |
| Project Name | Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies |
| Maximum Marks | 4 Marks |

# For Level Damage

BATCH\_SIZE = 32

# Data augmentation on training data

level\_train\_generator = train\_datagen.flow\_from\_directory(r'D:\Nalaiya Thiran Project\Juypter\dataset\Car damage\level\training',

target\_size = (224,224),

batch\_size = BATCH\_SIZE,

class\_mode = 'categorical')

# Data augmentation on testing data

level\_test\_generator = train\_datagen.flow\_from\_directory(r'D:\Nalaiya Thiran Project\Juypter\dataset\Car damage\level\validation',

target\_size = (224,224),

batch\_size = BATCH\_SIZE,

class\_mode = 'categorical')

# Model Building

# Importing required lib.

from tensorflow.keras.preprocessing import image

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

from tensorflow.keras.models import Model

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

from livelossplot import PlotLossesKeras

from keras.applications.vgg16 import VGG16

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

from glob import glob

import numpy as np

import matplotlib.pyplot as plt

# Building Model blocks

base\_model = VGG16(input\_shape = (224, 224, 3), # Shape of our images

include\_top = False, # Leave out the last fully connected layer

weights = 'imagenet')

# Since we don’t have to train all the layers, we make them non\_trainable:

for layer in base\_model.layers:

layer.trainable = False

# Compile

# Flatten the output layer to 1 dimension

x = Flatten()(base\_model.output)

# Add a final sigmoid layer with 1 node for classification output

x = Dense(3, activation='softmax')(x)

model = Model(base\_model.input, x)

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

model.summary()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

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

input\_3 (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

Total params: 14,789,955

Trainable params: 75,267

Non-trainable params: 14,714,688

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Model Fit

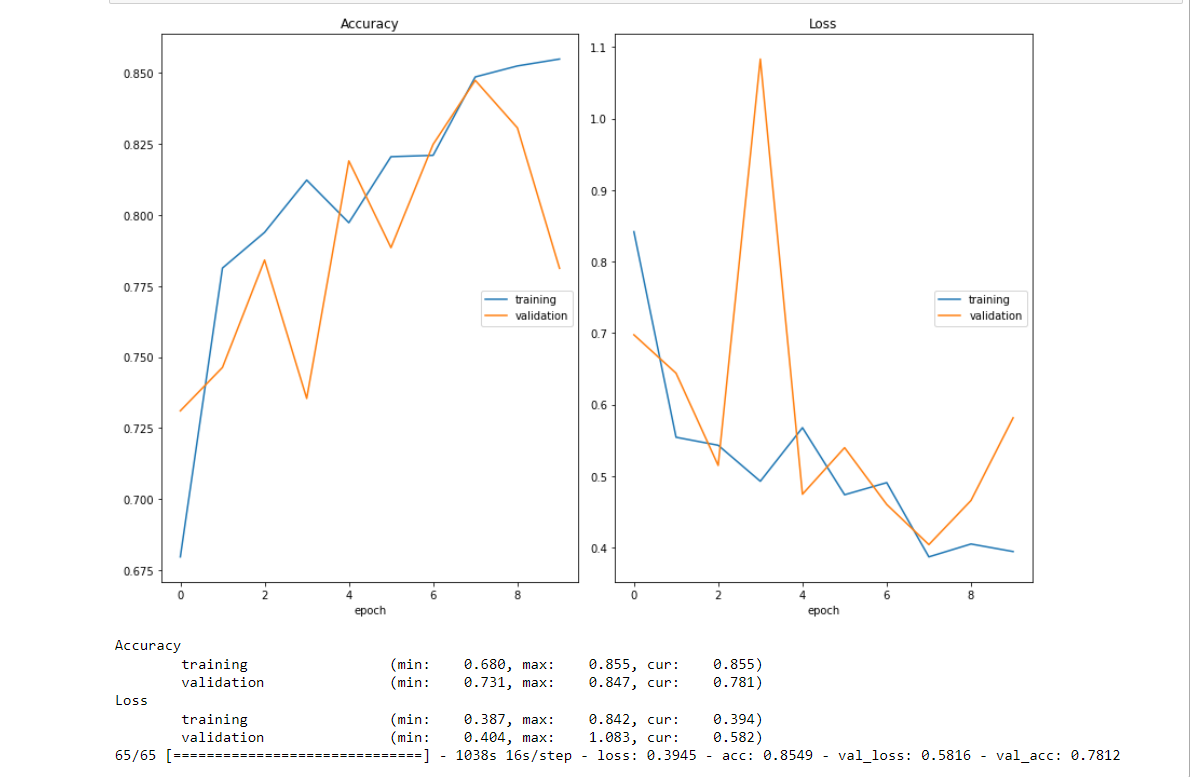
vggfit = model.fit(level\_train\_generator,

validation\_data = level\_test\_generator,

steps\_per\_epoch = 2099//BATCH\_SIZE,

validation\_steps = 1381//BATCH\_SIZE,

callbacks=[PlotLossesKeras()],

 epochs = 10)

Accuracy

training (min: 0.680, max: 0.855, cur: 0.855)

validation (min: 0.731, max: 0.847, cur: 0.781)

Loss

training (min: 0.387, max: 0.842, cur: 0.394)

validation (min: 0.404, max: 1.083, cur: 0.582)

# Save the Model

model.save('level.h5')

# Testing the Model

# Importing the required libraries

from tensorflow.keras.models import load\_model

import cv2

from skimage.transform import resize

#Loading the level model

model = load\_model('level.h5')

#Creating a function, which is used to predict the level

def detect(frame):

img = cv2.resize(frame,(224,224)) #Resize the image to model trained image size

img = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB) #Uploaded image is in the form of BGR, so convert to RGB

#Sacling need to be done

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

img = img/255.0

img = np.array([img]) # Doing array format

prediction = model.predict(img)

label = ["Minor", "Moderate", "Severe"]

preds = label[np.argmax(prediction)]

return preds

# Predictions

#Importing library

import numpy as np

# 1st testing

data = 'D:\\Nalaiya Thiran Project\\Juypter\\dataset\\Car damage\\level\\validation\\02-moderate\\0010.JPEG'

image = cv2.imread(data)

print(detect(image))

1/1 [==============================] - 0s 470ms/step

Minor