National College of Ireland

Configuration Manual

MSc Research Project
Master of Science in Data Analytics

Shubham Chaudhari
Student ID: x20160836

School of Computing
National College of Ireland

Supervisor: Prof. Mohammed Hasanuzzaman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shubham Sarjerao Chaudhari
Student ID:	x20160836
Programme:	MSc in Data Analytics **Year:** 2021-2022
Module:	MSc Research Project
Lecturer:	Prof. Mohammed Hasanuzzaman
Submission Due Date:	31th January 2022
Project Title:	Deep Learning Networks for Detection, Classification and Analysis of Car Damage
Word Count:	**1494** **Page Count: 16**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shubham Sarjerao Chaudhari

Date: 31th January 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shubham Chaudhari
Student ID: x20160836

1 Introduction

The prerequisites for installing the system, which was intended for categorization and detection of car damages utilizing Deep Learning models are outlined in the setup manual below. In addition, the document will go through the hardware as well as software requirements that were utilized to complete the project successfully.

2 System Configuration

The following is a list of the software and hardware configurations that were utilized to complete this project. The following are the hardware and software configurations utilized in the implementation:

2.1 Hardware Requirements

Table 1 depicts used hardware configuration to complete the project successfully.

Table 1 Hardware Requirements

Hardware	Configuration
System	ASUS ZenBook Duo
Operating System	Windows 10 (64bit)
RAM	16 GB
Hard Disk	1 TB (Solid State Drive)
Graphics Card	NVIDIA GeForce MTX250 (2GB)
Processor	Intel (R) Core (TM) i7
CUDA Version	11.3

This project was run on Windows 10, which has a 64-bit operating system. Installed RAM was 16 GB and installed GPU was NVIDIA GeForce MTX 250 2GB with cuda version 11.3, and the storage capacity was 1 TB Solid State Drive. Processor installed was Inter(R) Core (TM) i7.

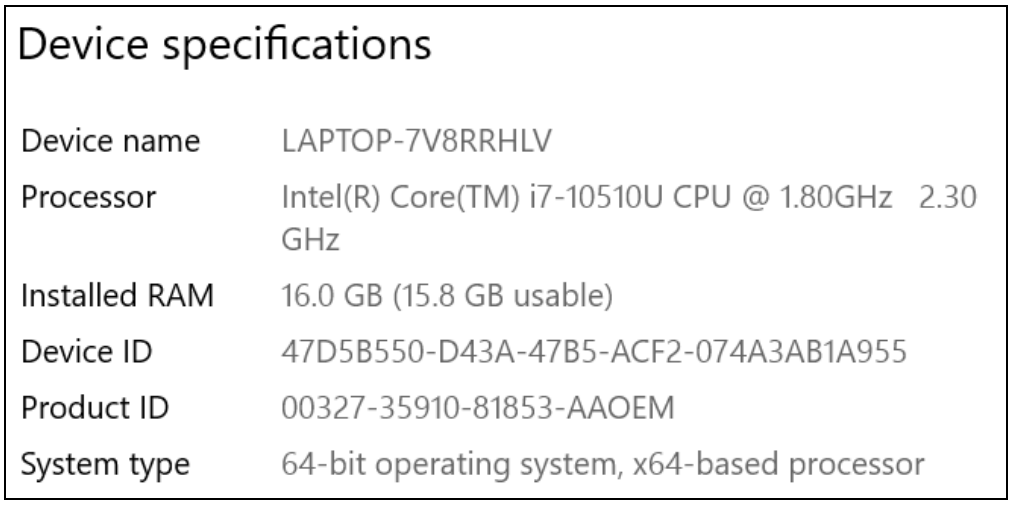

Figure 1 Laptop Configuration

2.2 Software Requirements

Table 1 depicts used hardware configuration to complete the project successfully.

Table 2 Software Requirements

Software	Configuration
Python	3.7 (64 bit)
Microsoft Excel	2019 Edition
Google Colab GPU	Tesla K80
Anaconda	1.10.0
VS Code	1.63.1

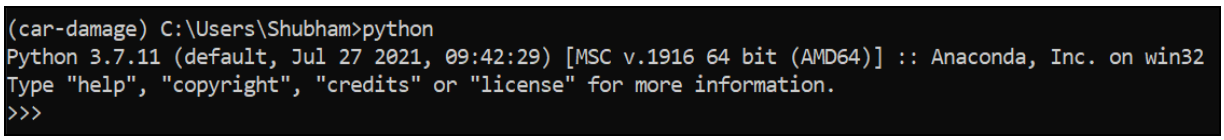

Figure 2 Python Configuration

Figure 3 Excel Configuration

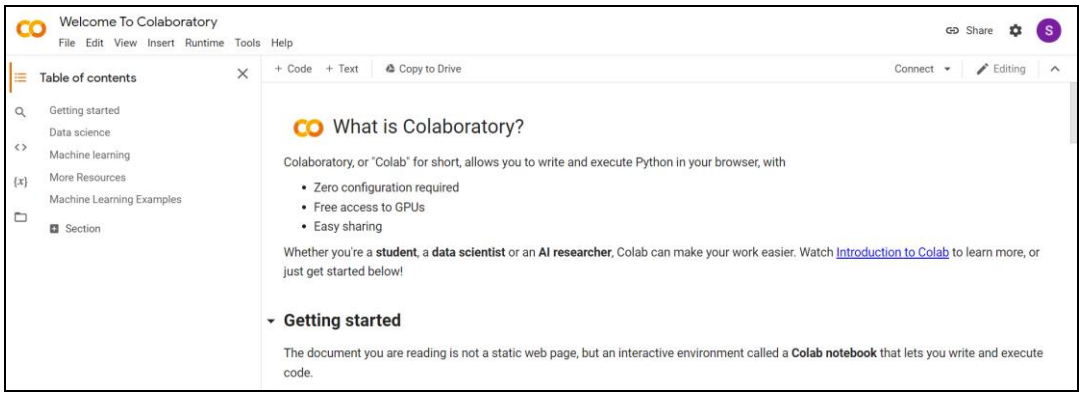

Figure 4 Google Colab Configuration

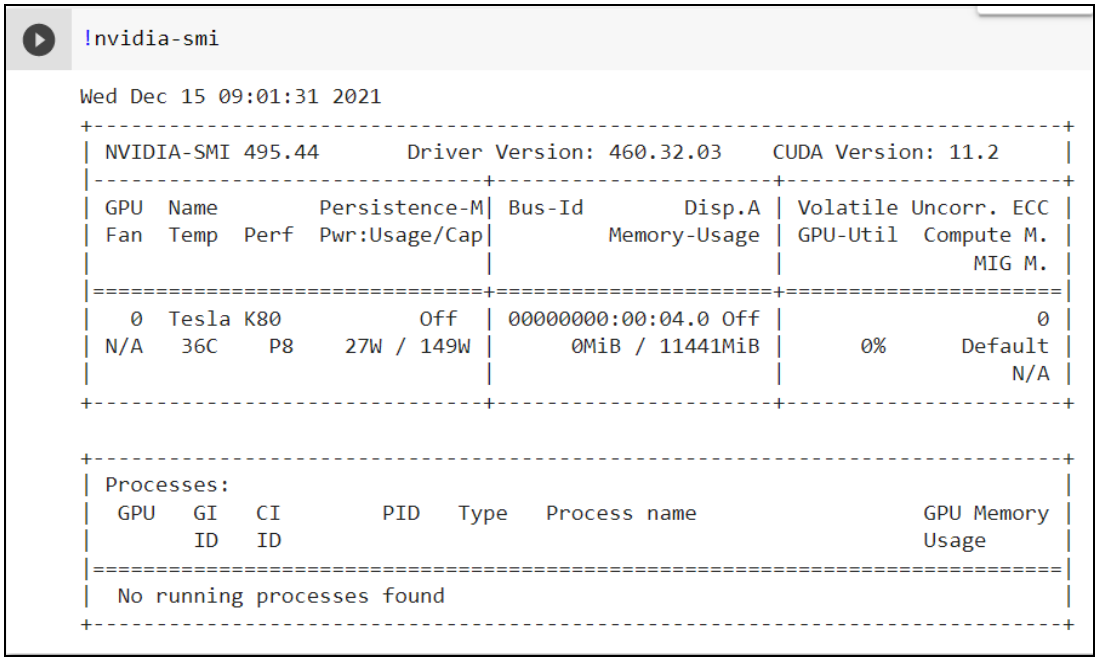

Figure 5 Google Colab GPU Configuration

For training High-End Deep Learning Neural Network models Tesla K80 GPU from Google Colab was used.

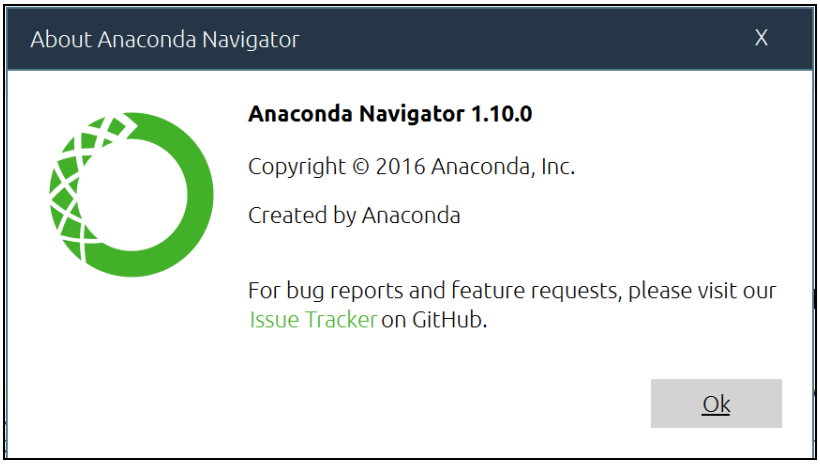

Figure 6 Anaconda Configuration

Anaconda Navigator 1.10.0 was used to create specific environments for completion of this project. Also, it was used to download necessary libraries.

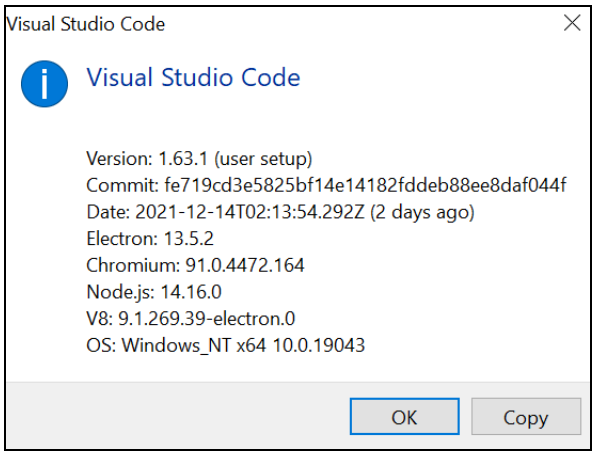

Figure 7 VS Code Configuration

VS-Code was used to create supporting as well as main python scripts for completion of the project.

3 Model Implementation

3.1 Data Collection

Step 1 – Creating an account with SerpAPI to scrap images from Google Images.
SerpAPI was utilised to scrape the data that was used in the implementation from Google. Initially, account need to be created on SerpAPI for further processing. Figure 8 shows dashboard of SerpAPI.

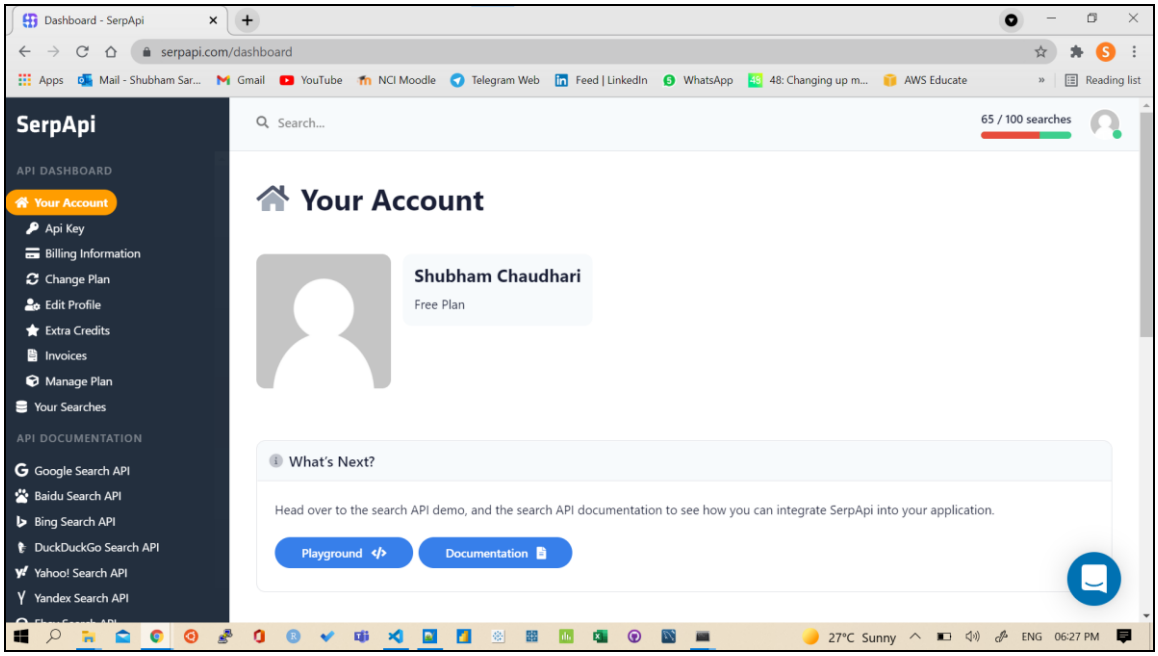

Figure 8 SerpAPI Dashboard

Figure 9 shows Google image API screen from which images were scrapped. Images were scrapped as per batches and each batch was responsible for fetching 100 images. So, if we want the first 500 images, we'll need to conduct five searches.

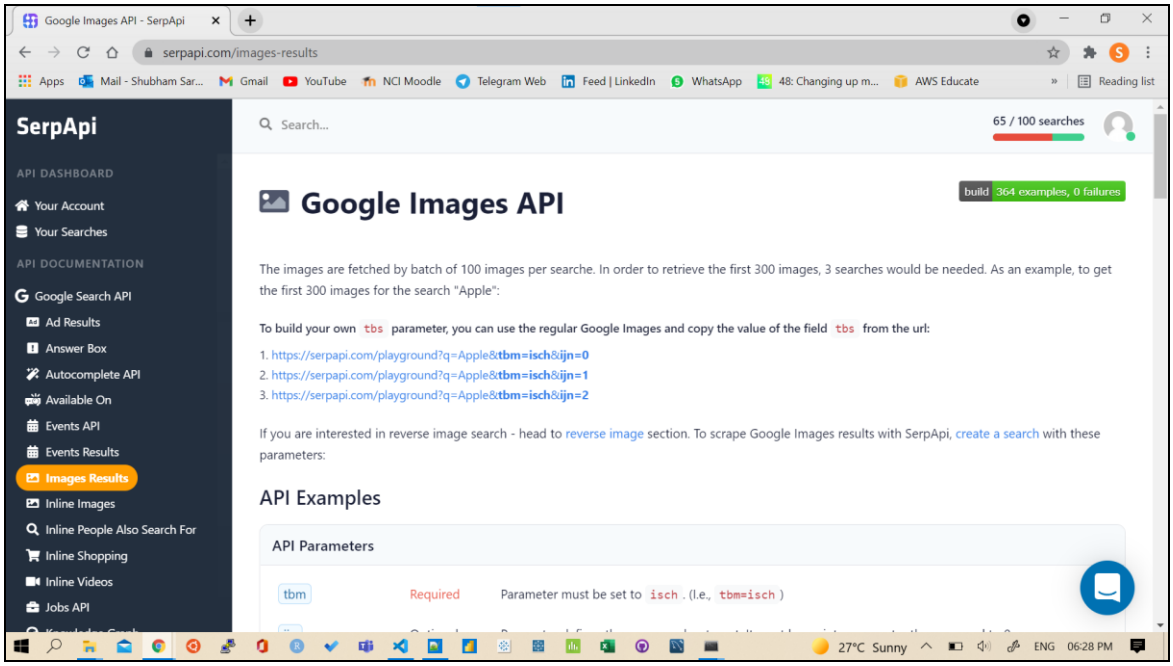

Figure 9 Google API Images

Figure 10 shows number of successful as well as failed searches per batch. From figure, we can conclude that number of successful searches are more than failed searches. It also shows statistics of API including number of searches per month, searches from account creation and so on.

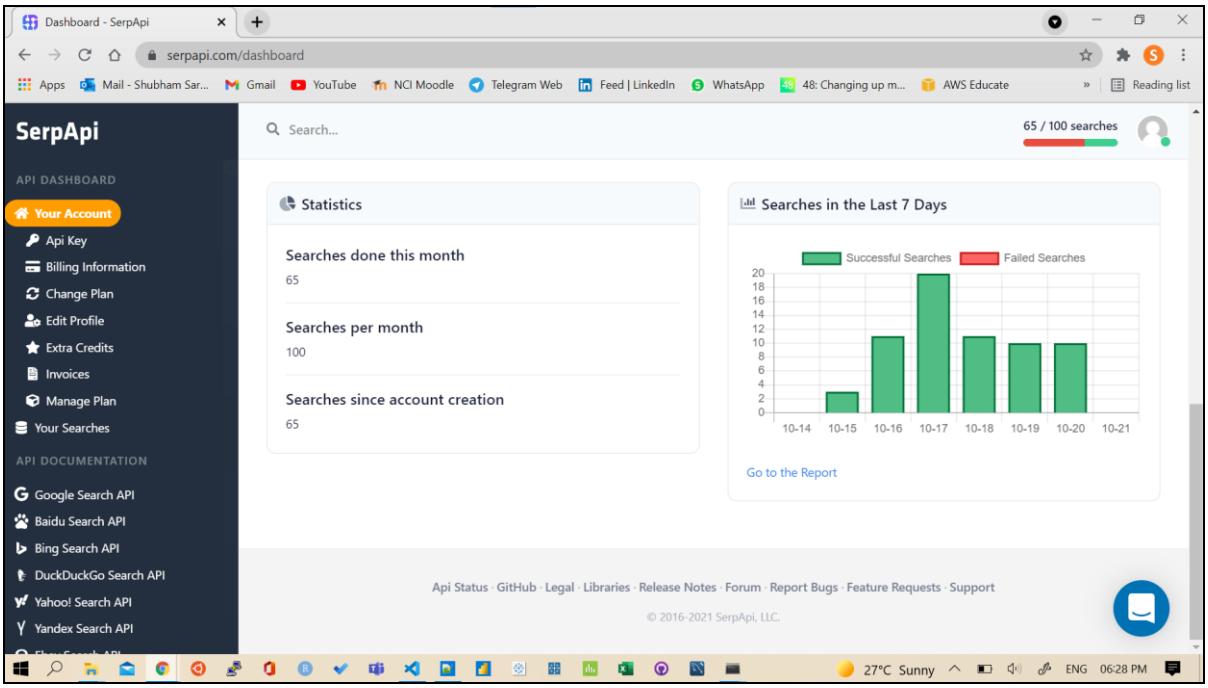

Figure 10 Account Usage

Step 2 – Execute “serp-api.py” to store image links in excel file.
Step 3 – Execute “image-downloader.py” to store image in different folders using excel file.

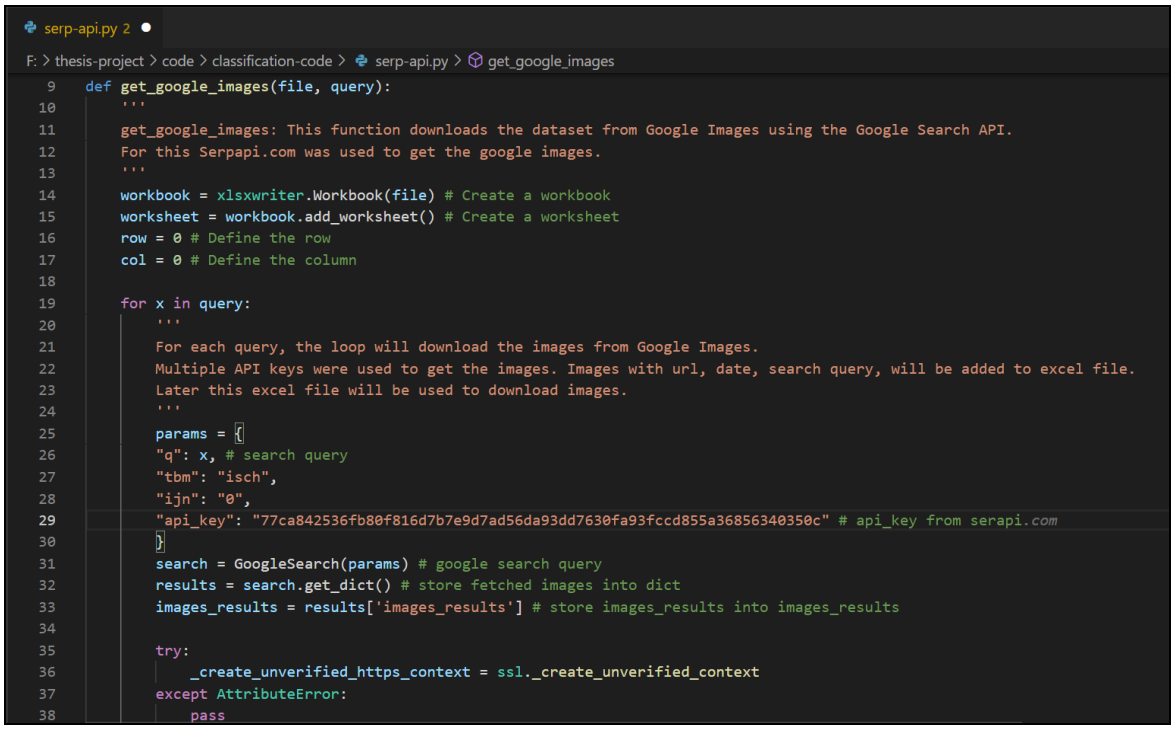

Figure 11 Excel File Creation with Dataset Link and Search Query

Function “get_google_images” allow to download and create excel file containing image links, search query, image downloaded date.

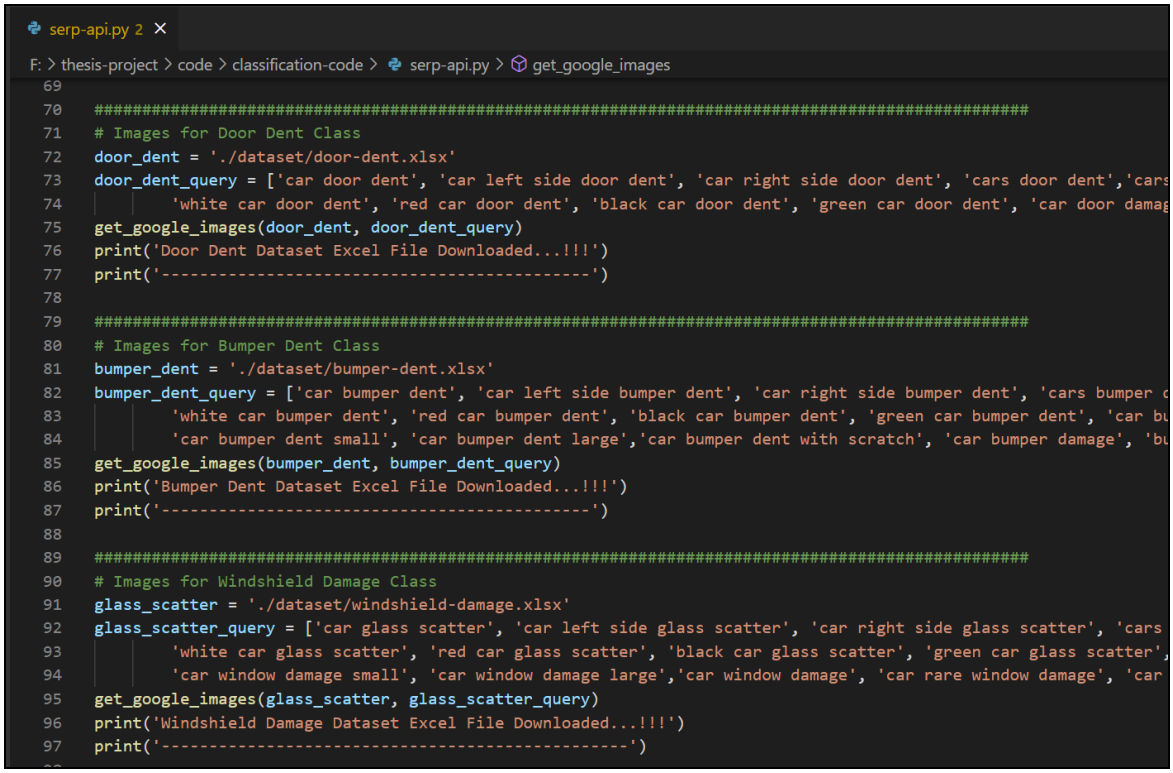

Figure 12 Search queries for different classes

Calling the function “get_google_images” with different search queries to get multiple results, and saving it into an excel file.

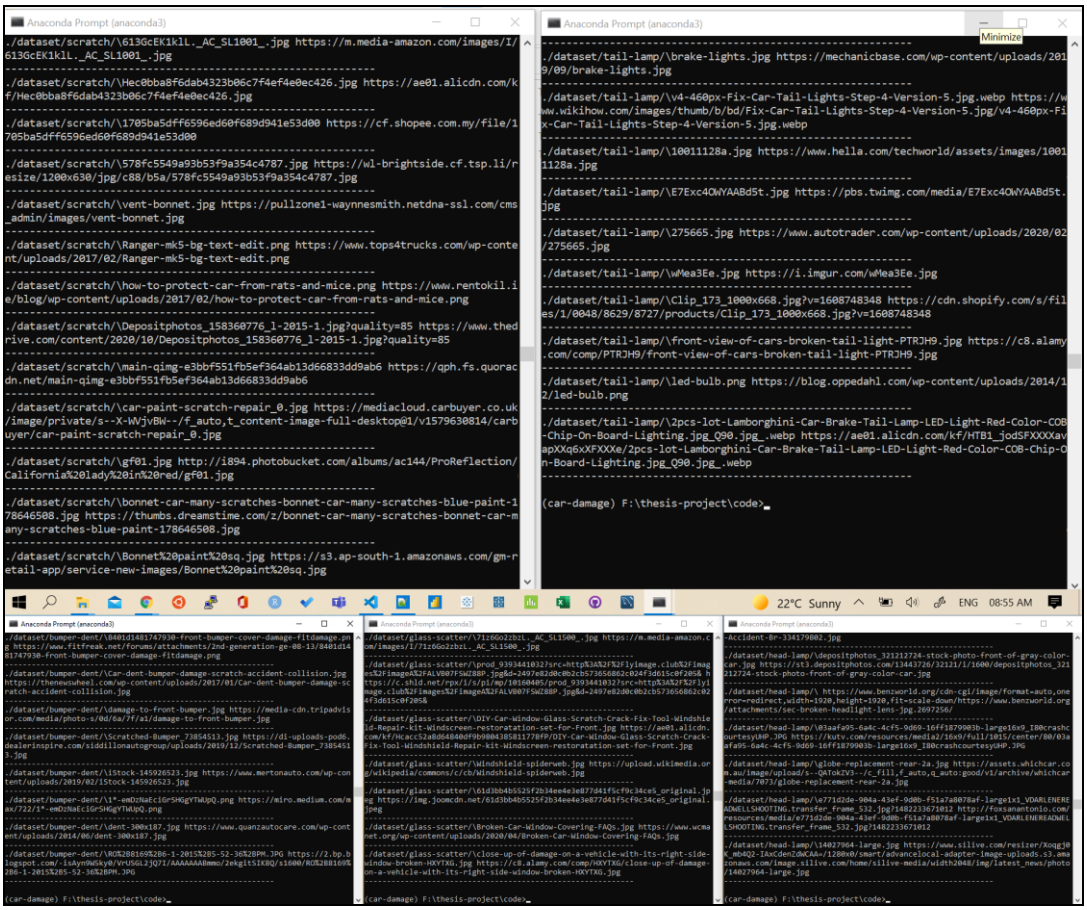

Figure 13 Downloading Images Execution

Figure 13 shows successfully downloaded images included with its path, search query as well as URL and store this meta-data into an excel file.

	A	B	C	D	E
1	image_name	image_url	date	search_query	
2	fix-car-dents-8-easy-ways-	https://img.wonderhowto.com/img/90/02/635724909525	Oct 21 2021 07:31 PM	car bumper dent	
3	bumper-dent-146385.jpg	https://cimg1.ibsrv.net/cimg/www.doityourself.com/660x	Oct 21 2021 07:31 PM	car bumper dent	
4	dented-car-bumper-use-a-	https://blog.masterappliance.com/wp-content/uploads/2	Oct 21 2021 07:31 PM	car bumper dent	
5	a-dent-in-a-car-KP90KD.jpg	https://c8.alamy.com/comp/KP90KD/a-dent-in-a-car-KP9C	Oct 21 2021 07:31 PM	car bumper dent	
6	car-dent.jpg	https://fixautousa.com/wp-content/uploads/2018/09/car	Oct 21 2021 07:31 PM	car bumper dent	
7	Paintless-Dent-Repair-Prist	https://pristinedentrepair.com/wp-content/uploads/2019	Oct 21 2021 07:31 PM	car bumper dent	
8	6a00d83451b17c69e20167	https://newpath.typepad.com/.a/6a00d83451b17c69e201	Oct 21 2021 07:31 PM	car bumper dent	
9	20150310_0858571.jpg	http://jerseydents.com/wp-content/uploads/2017/09/201	Oct 21 2021 07:31 PM	car bumper dent	
10	ToyotaCamryBumper.jpg	http://damndents.com/portals/40/img/Bumpers/ToyotaC	Oct 21 2021 07:31 PM	car bumper dent	
11	beforeandafterfebruary4.j	https://bumperbuddies.com/wp-content/uploads/2019/0	Oct 21 2021 07:31 PM	car bumper dent	
12	bumper-repair.jpg	https://louisvilledentrepair.com/wp-content/uploads/201	Oct 21 2021 07:31 PM	car bumper dent	
13	shutterstock_1106281790.	https://d32ptomnhiuevv.cloudfront.net/en-gb/sites/defa	Oct 21 2021 07:31 PM	car bumper dent	
14	ONfNVGPYSExxg_scVt43l6I	https://pristinedentrepair.com/wp-content/uploads/2019	Oct 21 2021 07:31 PM	car bumper dent	
15	AUTOCOLOR_PlasticBumpe	https://autocolorwi.com/wp-content/uploads/2020/10/A	Oct 21 2021 07:31 PM	car bumper dent	
16	A-dented-bumper-can-eas	https://blog.masterappliance.com/wp-content/uploads/2	Oct 21 2021 07:31 PM	car bumper dent	
17	how-to-fix-a-dent-in-your-	https://www.thesun.co.uk/wp-content/uploads/2017/02/	Oct 21 2021 07:31 PM	car bumper dent	
18	FP31V3OGM7E9XZ4.jpg?au	https://content.instructables.com/ORIG/FP3/1V3O/GM7E	Oct 21 2021 07:31 PM	car bumper dent	
19	car-bumper-dent-fixed-wit	https://s1.cdn.autoevolution.com/images/news/car-bump	Oct 21 2021 07:31 PM	car bumper dent	
20	dent-bumper-hack-700x35	https://inteng-storage.s3.amazonaws.com/images/uploac	Oct 21 2021 07:31 PM	car bumper dent	
21	maxresdefault.jpg	https://i.ytimg.com/vi/Ty3jG1JlHek/maxresdefault.jpg	Oct 21 2021 07:31 PM	car bumper dent	

Figure 15 Dataset Metadata Snapshot

3.2 Data Pre-processing

3.2.1 Image Resizing

Step 4 – Execute "image-resize.py" to resize all the images in the folders to 1024 X 1024. Image compression is the process of increasing or decreasing the size of an image without removing any content.

```
image-resize.py 1 ×
F: > thesis-project > code > classification-code > image-resize.py > ...
# Import Libraries
from PIL import Image
import glob
import os

# new folder path (may need to alter for Windows OS)
# change path to your path
ORI_PATH = 'F://thesis-project//code//dataset//cleaned-new-images//'
W = 1024
H = 1024
PATH = 'F://thesis-project//code//dataset//cleaned-new-images-resize//' #the path where to save resized images

# create new folder
if not os.path.exists(PATH):
    os.makedirs(PATH)

# loop over existing images and resize
# change path to your path
for filename in glob.glob(ORI_PATH+'**/*.*'): #path of raw images with is subdirectory
    img = Image.open(filename).resize((W,H))
    rgb_im = img.convert('RGB')

    # get the original location and find its subdir
    loc = os.path.split(filename)[0]
    subdir = loc.split('\\')[1]

    # assembly with its full new directory
    fullnew_subdir = PATH+"/"+subdir
    name = os.path.split(filename)[1]
```

Figure 16 Image Resize Code

```
Anaconda Prompt (anaconda3)
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************
Image Resized 1024  X  1024 Stored in - windshield-damage
***********************************************

(car-damage) F:\thesis-project\code>
```

Figure 17 Image Resize Execution

Our acquired car image data was of varying sizes; each image's pixel was unique, necessitating resizing to the same scale. Figure 17 shows code for resizing all augmented images into same scale i.e., 1024*1024. Here W and H represents width and height of image and converted images get stored into newly created folder with existing file name.
As a result, we transformed all images to a fixed size format of 1024 × 1024 pixels. Figure 10 shows resized image results of each damage category.

3.2.2 Image Augmentation for Damage Classification

Step 5 – Execute "image-augmentation.py" to increase the number of images within specific folder and store it into new folder.
Data augmentation is the process of increasing the amount of data required to train a classifier. Image-augmentation.py file was used for augmenting images for further classification training. Different types of functions were applied on image data like horizontal flipping as well as vertical flipping of images. Augmenter's library is used for image augmentation. Figure 18 shows code snippet of image augmentation. And figure 8 represents successful downloading of augmented images.

image-augmentation.py 7

F: > thesis-project > code > classification-code > image-augmentation.py > ...

```
12
13  images = [] # create a list to store all the images
14
15  # get all the image files from different classes by sepcifying the path
16  files = glob.glob("F://thesis-project//code//dataset//classification-dataset//cleaned-new-images-resize//tail-lamp//*.jpg")
17
18  # Stack Images
19  for myFile in files:
20      image = cv.imread (myFile) # read the image
21      image_rgb = cv.cvtColor(image, cv.COLOR_BGR2RGB) #change from BRG to RGB
22      images.append(image_rgb) # append the image to the list
23
24  sometimes = lambda aug: iaa.Sometimes(0.5, aug) # create a lambda function to apply the augmentation
25
26  # Define our sequence of augmentation steps that will be applied to every image.
27  seq = iaa.Sequential(
28      [
29          iaa.Fliplr(0.4), # horizontally flip 50% of all images
30          iaa.Flipud(0.1), # vertically flip 20% of all images
31
32          sometimes(iaa.Affine(
33              rotate=(-10, 10),
34          )),
35                  iaa.OneOf([
36                      iaa.GaussianBlur((0, 0.0))
37                  ])],
38
39          random_order=True
40  )
41
```

Figure 18 Image Augmentation Code

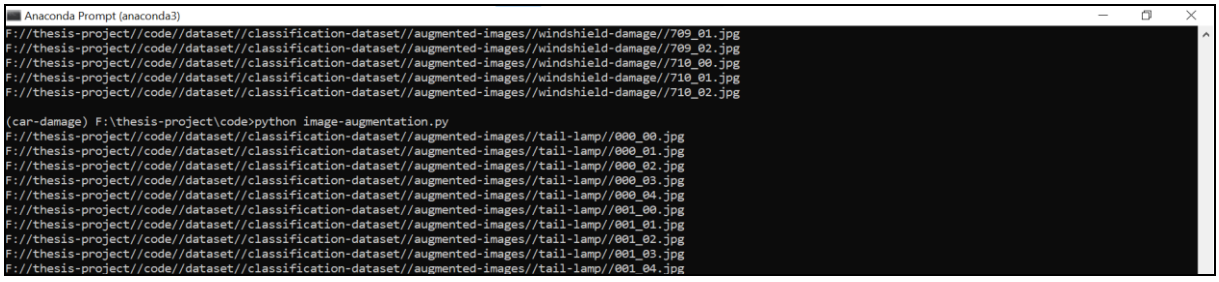

Figure 19 Image Augmentation Code Execution

3.2.3 Image Annotations for Damage Detection

Step 6 – Upload the original downloaded folder to CVAT by creating different projects with specific category to annotate. Select Bounding Box as annotation type for image annotations.

CVAT is a web-based image and video annotation tool that is free as well as open source. For image annotations, initially account need to be created on CVAT for further processing. Figure 20 represents dashboard for creating new task in CVAT.

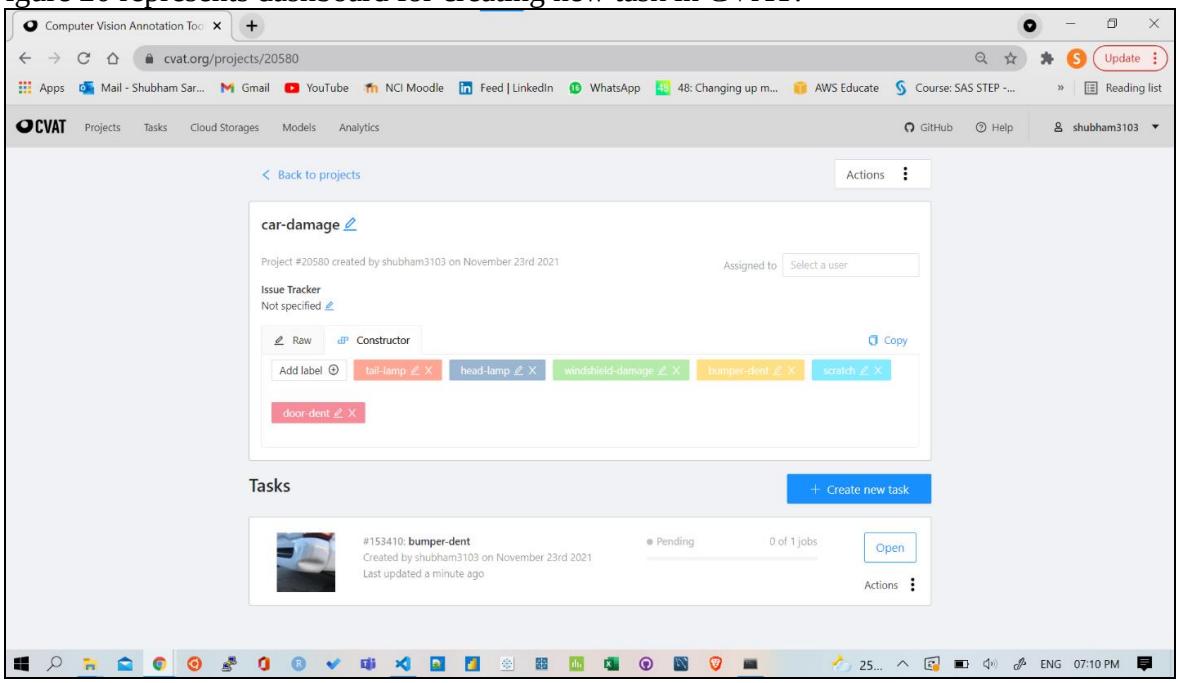

Figure 20 Task Creation on CVAT

In CVAT Tool task needs to be created prior starting the annotation process. In task multiple projects can be created with respect to classes. Classes and attributes need to be added Add Label section. Labels like Tail Lamp, Head Lamp, Windshield Damage, Bumper Dent, Scratch and Door Dent were added. Also, images were also uploaded in the same section.

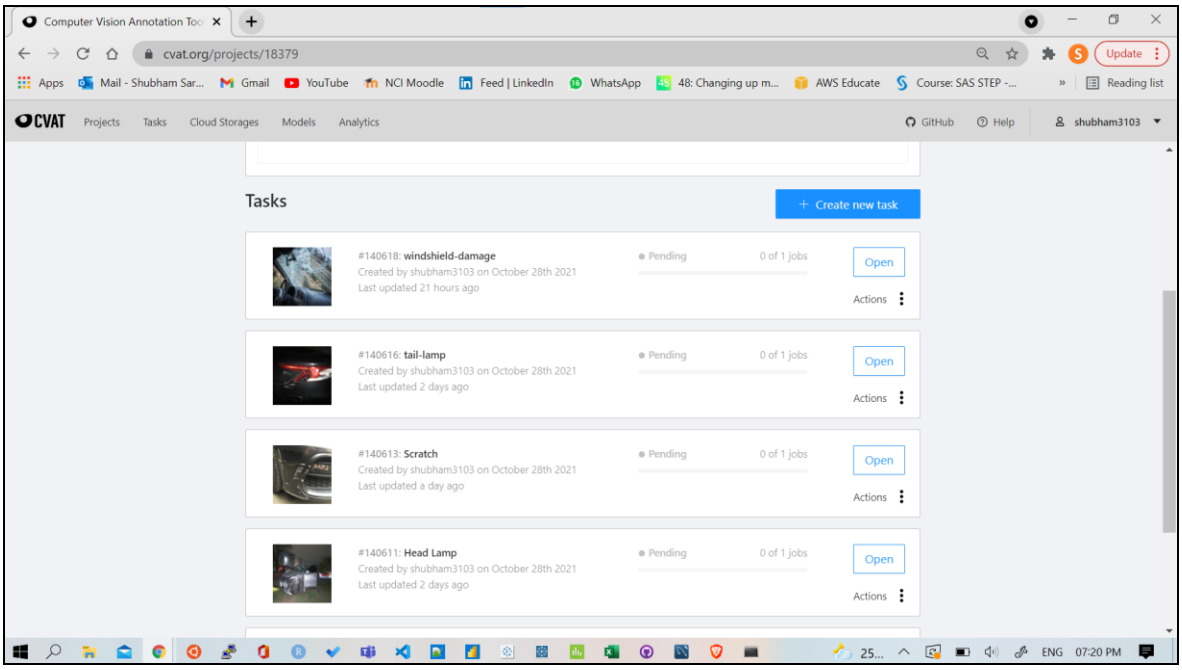

Figure 21 Different tasks according to classes

Different tasks were created according to different classes. Tasks like Windshield Damage Annotations, Tail Lamp Damage Annotations, Scratch Annotations, Bumper Dent Annotations, Door Dent Annotations and Head Lamp Damage Annotations were created. Opening a particular task gets to annotation page.

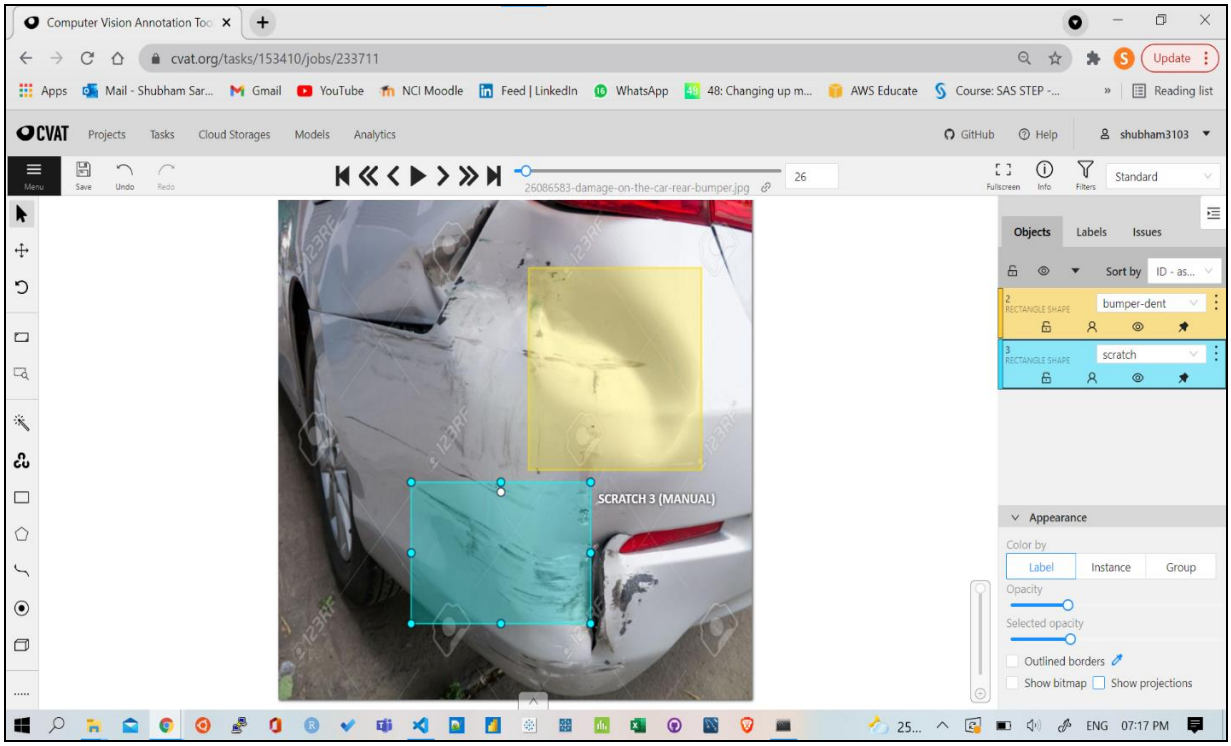

Figure 22 Annotation Dashboard

Figure 22 shows annotations dashboard. Here we can annotate with specific labels as well as with different annotations type. For the project Bounding box annotation type was selected.

3.3 Model Building

3.3.1 Damage Detection Model

Step 7 – Download all the annotated data into COCO format.
Step 8 – Execute "coco_json_to_csv.py" to convert COCO format file to CSV file.
Step 9 – Execute "plot_results.py" to check the bounding boxes positions on images after converting COCO JSON file to CSV.
Step 10 – Execute "retinanet.py" to train the model.
Step 11 – Upload the model weight file and test dataset on google drive for inferencing.
Step 12 – Execute "car-damage-detection.ipynb" to check the output results.

Object detection is important stage which has to be done before classification. RetinaNet is a well-known single-stage detector that is both accurate and fast. RetinaNet employs a feature pyramid network to recognize objects at several scales. To achieve damage detection objective, RetinaNet is used for implementing damage detection.

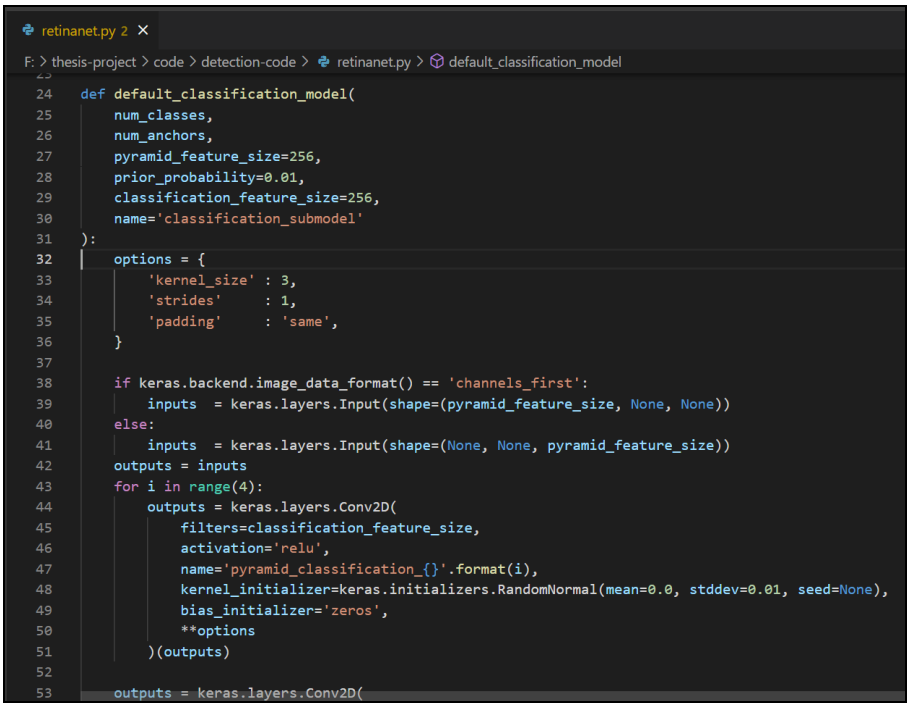

Figure 23 RetinaNet Model Code

Figure 24 RetinaNet inference code

3.3.2 Damage Classification Model

Step 13 – Execute "car-damage-classification.ipynb" to train different classification model as well as to inference the final output results.

After successful detection of damage, our next task was to classify that damage into correct category. For classification, various deep learning models were trained on pre-processed data. Figure 25 shows basic CNN model's layers.

```
class MyCNN():
    '''
    Class MyCNN : This class defines a basic Sequential CNN Model with Trainable params: 120,902
    It takes two function named __init__ and build.
    Activation function used is softmax
    '''
    def __init__(self, classes):
        self.classes = classes

    def build(self):
        model = Sequential() # Defining a model
        model.add(Convolution2D(32, (3, 3) , input_shape=(224,224,3))) # Input Layer
        model.add(MaxPooling2D(pool_size=(2, 2))) # Pooling Layer
        model.add(Dropout(0.5)) # Droput Layer
        model.add(Convolution2D(64, ( 3, 3))) # Hidden Layer
        model.add(MaxPooling2D(pool_size=(2, 2))) # Pooling Layer
        model.add(Dropout(0.5)) # Droput Layer
        model.add(Convolution2D(128, (3, 3))) # Hidden Layer
        model.add(MaxPooling2D(pool_size=(8, 8))) # Pooling Layer
        model.add(Dropout(0.5)) # Droput Layer
        model.add(Flatten()) # Flatten Layer
        model.add(Dense(self.classes)) # Dense Layer
        model.add(Activation('softmax')) # Activation Function
        return model
```

Figure 25 CNN Model Layers

```
[ ] # Building the model using build function
    cnn = MyCNN(classes=num_classes)
    cnn = cnn.build()

    # Compiliing the model
    cnn.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

    # Prinint Model Summary
    print(cnn.summary()) # Summary of the model

    # Saving Model Weights
    weight_path = '/content/drive/MyDrive/weights/mycnn' # weight path

    # Early stopping & checkpointing the best model in ../working dir & restoring that as our model for prediction
    cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
    cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')

    # Defining mycnn model
    results = cnn.fit(x_train,Y_train,
                      batch_size=batch_size,epochs=epochs,
                      verbose=1,
                      validation_data=(x_test,Y_test),
                      callbacks=[cb_checkpointer, cb_early_stopper])

    print(results.history.keys())
```

Figure 26 Method for calling CNN model

Figure 26 shows calling method of CNN model, which consist of different parameters like batch size, number of epochs, loss function and so on.

After training basic CNN model, research is further processed by training same data on pre-trained models. Figure 27 represents creation of initial layers for pre-trained model training.

```python
# Create initial layers for training pre-trained models
class MyModel():
    '''
    class MyModel : This class defines top layers of pretrained models. This allows us to makes changes
    in layers at the top to be trained on customized dataset.
    '''
    def __init__(self, baseModel, classes, D):
        self.baseModel = baseModel
        self.classes = classes
        self.D = D

    def build(self):
        headModel = self.baseModel.output # Getting output from base model
        headModel = Flatten(name="Flatten")(headModel) # Flatten Layer
        headModel = Dense(self.D, activation='relu')(headModel) # Dense Layer
        headModel = Dropout(0.5)(headModel) # Dropout Layer
        headModel = Dense(self.classes, activation='softmax')(headModel) # Activation Function
        return headModel
```

Figure 27 Setting initial layers for pre-trained models

Some pretrained models were trained on pre-processed data for further assessment. Figure 28 shows training parameters of MobileNet model. Hyper parameters used while training MobileNet includes batch size i.e., 50, number of epochs, weights of pre-trained imagenet dataset, loss function, number of classes and so on.

```python
# MobileNet
# Model Hyperparameters
batch_size = 50 # Batch Size
epochs = 20 # Number of Epochs

# Load pretrained model:
baseModel = MobileNet(weights='imagenet', include_top=False,input_tensor=Input(shape=(224, 224, 3)))

# Initialize head of network
mobilenet = MyModel(baseModel, classes=num_classes, D=256)
headModel = mobilenet.build()

# Replace the first layers with headModel:
mobilenet = Model(inputs=baseModel.input, outputs=headModel)

# Unfreezing some of conv layers:
for layer in baseModel.layers[15:]:
    layer.trainable = True

# Complie pretrained mobilenet model
mobilenet.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

# Print summary of mobilenet model
print(mobilenet.summary())

# Trainned model weight path to be saved
weight_path = '/content/drive/MyDrive/weights/mobilenet/'

# Early stopping & checkpointing the best model in ../working dir & restoring that as our model for prediction
cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')
```

Figure 28 Training Mobile Net

Similar to the MobileNet, next model was trained that is nothing but VGG16. Figure 17 shows implementation and training of VGG16 model on same dataset. Parameters that are used while training VGg16 model includes number of parameters being freeze while training, batch size of images, number of epochs to be trained, loss function, initial layers that are replaced with base model, model summary and so on.

```
[ ]  # VGG16
     # Model Hyperparameters
     batch_size = 50 # Batch Size
     epochs = 20 # Number of Epochs

     # Load pretrained model:
     baseModel = VGG16(weights='imagenet', include_top=False,input_tensor=Input(shape=(224, 224, 3)))

     # Initialize head of network
     vgg16 = MyModel(baseModel, classes=num_classes, D=64)
     headModel = vgg16.build()

     # Replace the first layers with headModel:
     vgg16 = Model(inputs=baseModel.input, outputs=headModel)

     # Unfreezing some of conv layers:
     for layer in baseModel.layers[15:]:
         layer.trainable = True

     # Complie pretrained vgg16 model
     vgg16.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

     # Print summary of vgg16 model
     print(vgg16.summary())

     # Trainned model weight path to be saved
     weight_path = '/content/drive/MyDrive/weights/vgg16/'

     # Early stopping & checkpointing the best model in ../working dir & restoring that as our model for prediction
     cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
     cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')
```

Figure 29 Training VGG16

The Inception-ResNet-v2 structure is a convolutional neural network that is based upon this Inception group of models but includes residual connections. Figure 18 shows implementation snippet of InceptionResNetV2 model. Similar to other pre-trained models, this model also contains same parameters like base model for initial layers, loss function, weights and so on.

```
     # InceptionResNetV2
     # Load pretrained model:
     baseModel = InceptionResNetV2(weights='imagenet', include_top=False,input_tensor=Input(shape=(224, 224, 3)))

     # Initialize head of network
     inceptionresnetv2 = MyModel(baseModel, classes=num_classes, D=256)
     headModel = inceptionresnetv2.build()

     # Replace the first layers with headModel:
     inceptionresnetv2 = Model(inputs=baseModel.input, outputs=headModel)

     # Unfreezing some of conv layers:
     for layer in baseModel.layers[15:]:
         layer.trainable = True

     # Complie pretrained inceptionresnetv2 model
     inceptionresnetv2.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

     # Print summary of mobilenet model
     print(inceptionresnetv2.summary())

     # Trainned model weight path to be saved
     weight_path = '/content/drive/MyDrive/weights/inceptionresnetv2/'

     # Early stopping & checkpointing the best model in ../working dir & restoring that as our model for prediction
     cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = early_stop_patience)
     cb_checkpointer = ModelCheckpoint(filepath = weight_path, monitor = 'val_loss', save_best_only = True, mode = 'auto')

     # Fit the model created
     results = inceptionresnetv2.fit(x_train,Y_train,
                         batch_size=batch_size,epochs=epochs,
```

Figure 30 Training InceptionResnetv2

```
test_dir = test_path # Test dataset directory
images = [] # Empty list for images
titles = [] # Empty list for image lables

for img in glob.glob(test_dir + "*.jpg"):
    start_time = time.time() # Defining start timing for inferencing
    img_name = os.path.split(img)[-1]
    im = cv2.imread(img)
    im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
    img_og = image.load_img(img,target_size=(224,224))
    img = np.asarray(img_og)
    img = np.expand_dims(img, axis=0)
    prediction = inceptionresnetv2.predict(img) # Predicting image lables using inceptionresnetv2 Model
    MaxPosition=np.argmax(prediction)
    prediction_label=labels[MaxPosition]
    print('************************************************************')
    print(prediction_label)
    '''
    Writing predicted label on image for better visibility
    '''
    org = (50, 50)
    fontScale = 1
    color = (255, 0, 0)
    thickness = 2
    cv2.putText(im, prediction_label, org, cv2.FONT_HERSHEY_SIMPLEX,
                    fontScale, color, thickness, cv2.LINE_AA)
    end_time = time.time() # End time for prediction
    infer_timne = end_time - start_time
    print('Time Requirted: ',{infer_timne}) # Print total time required for making predictions on image
    plt.imshow(im)
    plt.show() # Plot image with predicted label
cv2.destroyAllWindows()
```

Figure 31 Model Inference Code

Figure 31 shows code snap for different model inferencing on different models trained.

References

Kyu, P.M. and Woraratpanya, K., 2020, July. Car Damage Detection and Classification. In Proceedings of the 11th International Conference on Advances in Information Technology (pp. 1-6).

Malik, H.S., Dwivedi, M., Omakar, S.N., Samal, S.R., Rathi, A., Monis, E.B., Khanna, B. and Tiwari, A., 2020. Deep Learning Based Car Damage Classification and Detection. EasyChair Preprint, (3008).

Patil, K., Kulkarni, M., Sriraman, A. and Karande, S., 2017, December. Deep learning based car damage classification. In 2017 16th IEEE international conference on machine learning and applications (ICMLA) (pp. 50-54). IEEE.