Final Project Report for Computer Vision

5435-(FB) Fall 2019

Topic: Object detection (Image and video)

Group members:

Jeevan Reddy Gopu (1105340)

Prashanth Kumar Daram (1116540)

Vishnu Sudheer Gannamani (1110200)

Instructor: Dr. Shan Du

Table of contents

Names	Page No
Abstract	3
Existing System and Proposed System	4
Advantages of YOLO	6
Advantages of CNN	7
Advantages of Fast RCNN	8
Execution Steps	8-16
Functions and Methods	16-18
Code and Result Screen Output	19-24
Applications	25
Similar Works	26
Regression and classification	27
YOLO Algorithm	27-29
CNN Algorithm	29-31
Training Data	31
Design	31-32
System Test	32-33
Block Diagram	33-35
UML Diagrams	36-39
System Requirements	40
Conclusion	41

Abstract:

The project Which we are dealing is based on mainly object detection. Detecting the objects that are existing in the video and audio. The accuracy of determining these objects will be high and that is the main goal of the project. There is a significant difference in between the previous one and the existing one which we used. The previous methods of object detection were based on older techniques and they used some classifiers and other methods such as splitting the image into parts then extracting features and working. The latest methods have been developed to a lot of extent and now doing object detection is very easy and always accessible. Using classifiers to detect objects is old fashion. Object detection is now considered as regression. The bounding boxes are generated around the thing and probabilities of class are named and labelled. A neural network is used to detect the objects at a time considering the entire image briefly and each part of it has a particular usage and function. The labels are identified for a set of classes. It is always optimizable always from any ending. The architecture which we used is very speed and unified way. YOLO is used in our project. It is a latest algorithm that is used to detect the objects. There are many versions of it, and we used the latest YOLOV3 version. It stands for" you look only once". It has a faster processing time and does 155 frames per second. It acts in a real way to detect images. Better than R-CNN. The language in which we wrote the code is python. Deep learning algorithms such as CNN, faster CNN and other were used. CNN stands for convolution neural network and it process the image into frames and gives better output. We did the object detection both for a video and an audio with high accuracy. Took a larger data set and lot of examples have been implemented. The code runs perfectly with no errors. Darknet is used to detect multiple objects.

Existing System:

The existing system uses many traditional methods for detecting the objects. It contains a various stream of pipeline process to complete the whole process. The traditional methods include the division of image into smaller regions and then performing algorithms on them and identifying classifiers and then processes the image. This method is slow and less accurate results. Scale invariant feature transform (SIFT) and HOG methods were used. These are some of the machine learning algorithms. SIFT uses bits of images and extracts features. Feature extraction algorithms were less worthy in them. Pipeline stages are high in them. Bounding boxes are not properly generated and classifiers were not properly identified and labelled. Multiple objects couldn't be identified at a single time. Training the data would be hard

Proposed system:

With the introduction of python and deep learning techniques there was a heavy growth in object detection concept. Python gave a lot of scope to object detection concept and made it easier to code for object detection. Latest technologies such as yolo version 3 and deep learning algorithms such as CNN, RCNN, fast R-CNN, single shot detection are some that took object detection to higher level. By Python The number of lines required to write the code has been reduced. Yolo has made implemented more because it takes the whole image at a time as input and processes the whole image in a single glance. Bounding boxes are generated accurately to all objects in an image. Classification is done at a higher level with great accuracy. Our proposed project is the image is taken as input and yolo and deep learning algorithms have been implemented and bounding boxes have been generated and classifiers work perfectly and labelled with the type detected and accuracy percentage is shown. Higher precision is implemented. Larger data set outputs have been used with lot of examples implemented. Video detection is also done in the project where we detect objects in the video using the same procedure and we identified moving objects in a video with same procedure used for images. CNN is used to divide object into sub layers and detect features individually

We implemented an algorithm from an IEEE Paper We choose the paper because:

The IEEE paper which we chose to do our project is based on YOlO, real time object detection using the deep learning algorithms. Written by four authors, **Joseph Redmon** (university of Washington). **Santosh Divvala** (Allen Institute of Artificial Intelligence, **Ross Girshick**(Facebook AI Research), **Ali Farhadi(University of Washington).** We choose this paper because here the unified architecture is extremely fast. Prior experiments were using classifiers to detect the objects. The experiment in this paper uses a single neural network to predict the bounding boxes and classification of various things in the experiment. The whole experiment runs under the single pipeline. We can have an end to end access. Performance can be detected in all the corners end to end. The authors of the paper detected the process as a regression problem to spatially separating bounding boxes that are linked with the probabilities of the class. The architecture they used was extremely fast. They took YOLO as their base model, and it processed 45 frames per second. They used fast YOLO to achieve 155 frames per second. Yolo has a problem of localisation errors but less. False detections won’t occur in that. Dpm and RCNN are less powerful than this. Humans vision is fast, and they detect in a glance. Deformable parts model is not used because it uses sliding window on various parts and results are not accurate. RCNN also uses bounding boxes and then run classifier and predict. This is quite slow but better compared to traditional methods. So, we used just used the roots of RCNN concepts. RCNN we used slightly because it imagines dark patches as objects. **The main reasons we choose this paper is because**: the algorithm used in the paper focuses on unified detection. Entire features are used for detection. Concentrates on full image and all objects. The image is divided into n number of grids. The bounding boxes are generated to each bounding cell and classifies. Five parameters such as x,y, width, height and confidence are taken into consideration. Conditional probabilities are used for perfect recognition in every unit cell. We were inspired by the CNN algorithm because it uses multiple layers in an image for detection. All the parameters are pretrained and the weights and labels are taken into consideration by implementing the algorithm on huge set of images and videos. Larger data sets were used by us to train the model in order to recognise. Learning rate is high for our algorithm. We had a small problem regarding very minute things in an image. We have chosen this paper because we felt this paper was best and they clearly mentioned which algorithms are more effective and what errors occurs if we use some methods. The training methodologies, the learning rate problems, the grid generation and how to implement at what times, the methods, effects, and the results were clearly explained. Comparison was done between various algorithms and their performance was presented along with graph analysis too. The defects in few algorithms and advantages in few algos were clearly depicted. State of the art facilities were conveyed properly. By taking consideration of all these YOLO, Deep learning methods such as CNN was used by us and we also took a little consideration of RCNN. This paper was very helpful to us in choosing the project. The paper solved many of our doubts and gave guidance to choose the best possible algorithms to get high outputs.

Paper link: https://pjreddie.com/media/files/papers/yolo.pdf

We used some existing algorithms from open source, we are presenting why we took that and why not others below

Darknet is used by us in our project. It is an open source algorithm that can be used for object detection. It is an open source neural network. It is fast version and it also supports GPU. It is linked with Yolo. CNN works well to detect one image that is centred in the image and if we have so many objects in the image and we want to detect all at a single time Darknet yolo is the best software to do that. It may sometime mis-identify few objects. Dark objects may sometime be misjudged. It resizes the image and run the convolution network and does the non max suppression. Darknet is mainly written in C language and can be used in python too. Dark flow is the concept that we can call directly from python in order to perform these operations. We mainly used this concept because of multiple object detection and the other algorithms doesn't have this power to detect and the accuracy levels and time taken were also high.

c) We proposed an algorithm, The novelty and the advantages of it are mentioned below

The algorithm we proposed was developed in python and it became easy because of the programming language python. The algorithm which we proposed is latest and we used the advance technologies which were based on deep learning methods and implementation. Our algorithm had created high accuracy and it runs quicker to detect the objects. The amount of accuracy which we obtained is higher for the detected objects. The classifier we used is powerful and accurate. Implementation of our algorithm was simple and precise. The algorithms which we used consists of YOLO version 3 which is fast and advance, and it is the latest version of it. CNN was used at a higher level on the images. The code was contained in two python files. In one of the files we used the image and video was taken as input and spitted into n X n rows and columns. A file called weights is involved in it where it has lot of trained data with so many accurate features. All the grids are checked, and features are extracted from the grids that are present in the individual boxes. All the grids have an individual bounding box that has a measurement. The algorithm generates multi colours to the bounding boxes and depending on the object size and shape it generates that required dimension box around the image. All the grids are combined and checked with their respective weights for generating results and identifying to what class it is belonged to and we are labelling them with the accuracy percentage. We can feed the algorithm with any input and it has the capacity to detect most of the objects present in it. The algorithm clubs the various aspects to be covered and checks the multi-dimensional arrays matching features to check the

Advantages of YOLO

- Speed is 45 frames per second. Faster than Realtime
- Highly recommended for real time processing.
- The network understands the real-world images
- Smaller architecture
- Open source
- Full Image detection at a time.

Advantages of CNN

- Automatic detection of important features
- No human surveillance or intervention required
- Automatically it learns features of the images
- Parameters are shared. Computations and pooling are done
- It is universal and runs on many environments
- Multi layer feature extraction and pooling
- Layer wise applications and functions. Minute things are also detected

Advantages of fast RCNN

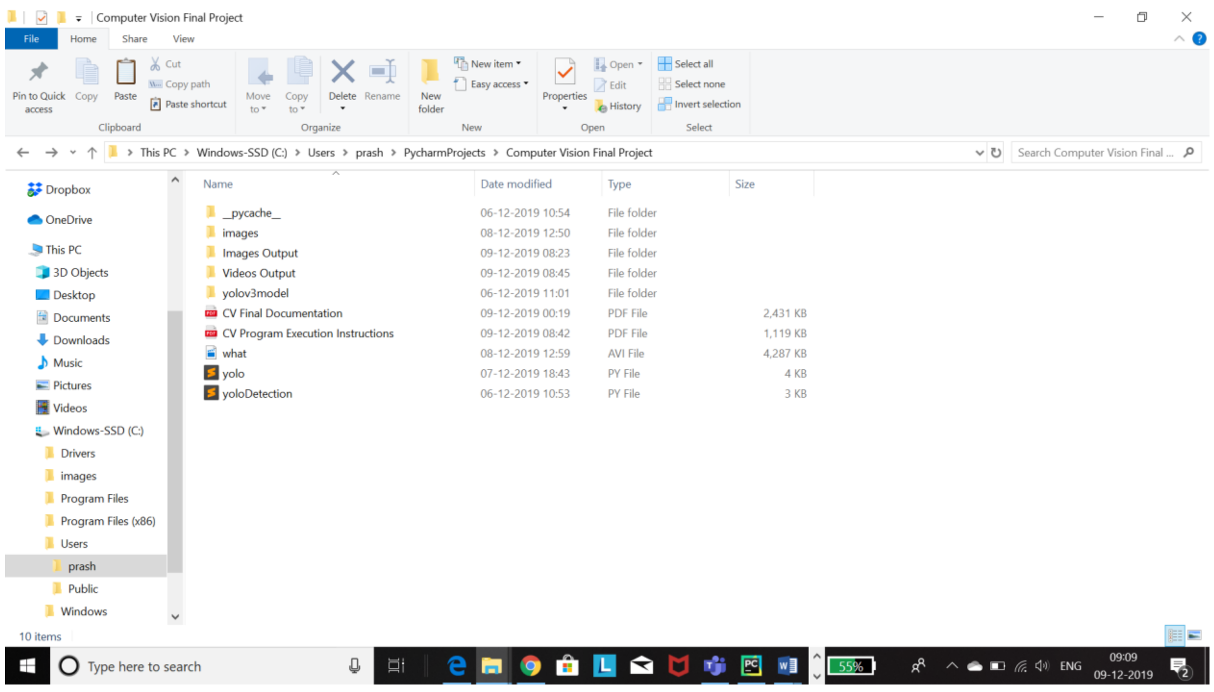
- Region based detection
- Improving local features of image
- Layering the image

4) Describe the procedure of your method in detail and show some results STEP by STEP.

The image and video is taken as input and all the places which contains objects are focussed and image is splitted into layers and pooling is done and detected and shown. Based on image length height width aspect ratio are calculated for objects and bounding boxes are drawn and it checks and matchjes with the previously extracted weights and features and classifiers labels the image with accuracy. Functions are applied on that to run this whole process.

Steps to Execute the program:

1.) Inorder to run this program, make sure we have all the required individual folders like _pycache, images, yolov3model, yolo.py and yoloDetection.py files in one single folder as shown in the below screenshot.

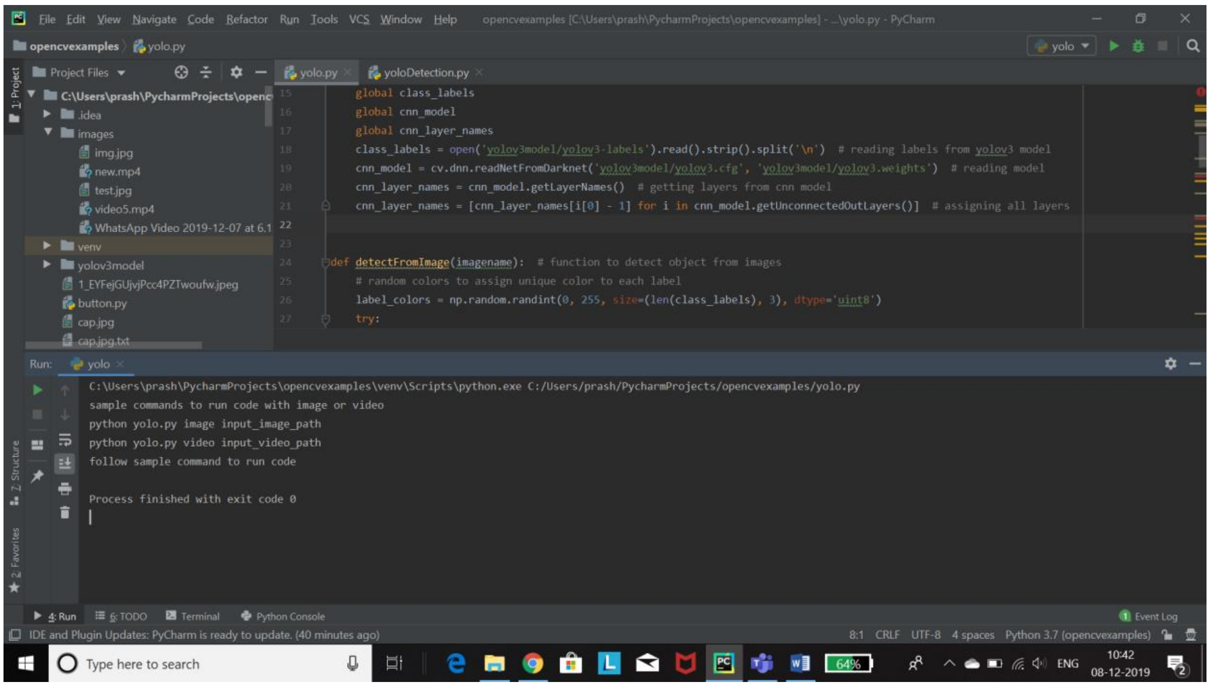

Computer Vision Final Project

This PC > Windows-SSD (C:) > Users > prash > PycharmProjects > Computer Vision Final Project

Name	Date modified	Type	Size
pycache	06-12-2019 10:54	File folder	
images	08-12-2019 12:50	File folder	
Images Output	09-12-2019 08:23	File folder	
Videos Output	09-12-2019 08:45	File folder	
yolov3model	06-12-2019 11:01	File folder	
CV Final Documentation	09-12-2019 00:19	PDF File	2,431 KB
CV Program Execution Instructions	09-12-2019 08:42	PDF File	1,119 KB
what	08-12-2019 12:59	AVI File	4,287 KB
yolo	07-12-2019 18:43	PY File	4 KB
yoloDetection	06-12-2019 10:53	PY File	3 KB

10 items

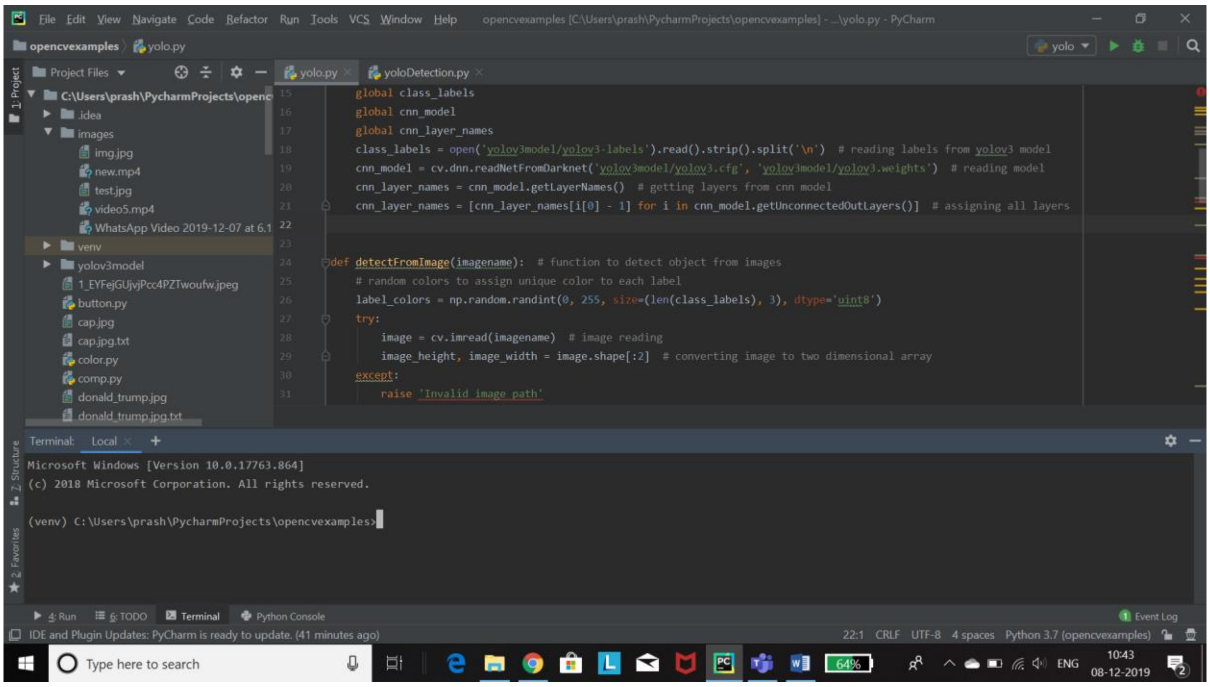
2.) In Step 2 We need to run the yolo.py file and make sure we get **“Process finished with exit code 0”** as shown in the below screenshot.



```
    global class_labels
    global cnn_model
    global cnn_layer_names
    class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n')  # reading labels from yolov3 model
    cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights')  # reading model
    cnn_layer_names = cnn_model.getLayerNames()  # getting layers from cnn model
    cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in cnn_model.getUnconnectedOutLayers()]  # assigning all layers


def detectFromImage(imagename):  # function to detect object from images
    # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3), dtype='uint8')
    try:
```

```
C:\Users\prash\PycharmProjects\opencvexamples\venv\Scripts\python.exe C:/Users/prash/PycharmProjects/opencvexamples/yolo.py
sample commands to run code with image or video
python yolo.py image input_image_path
python yolo.py video input_video_path
follow sample command to run code

Process finished with exit code 0
```

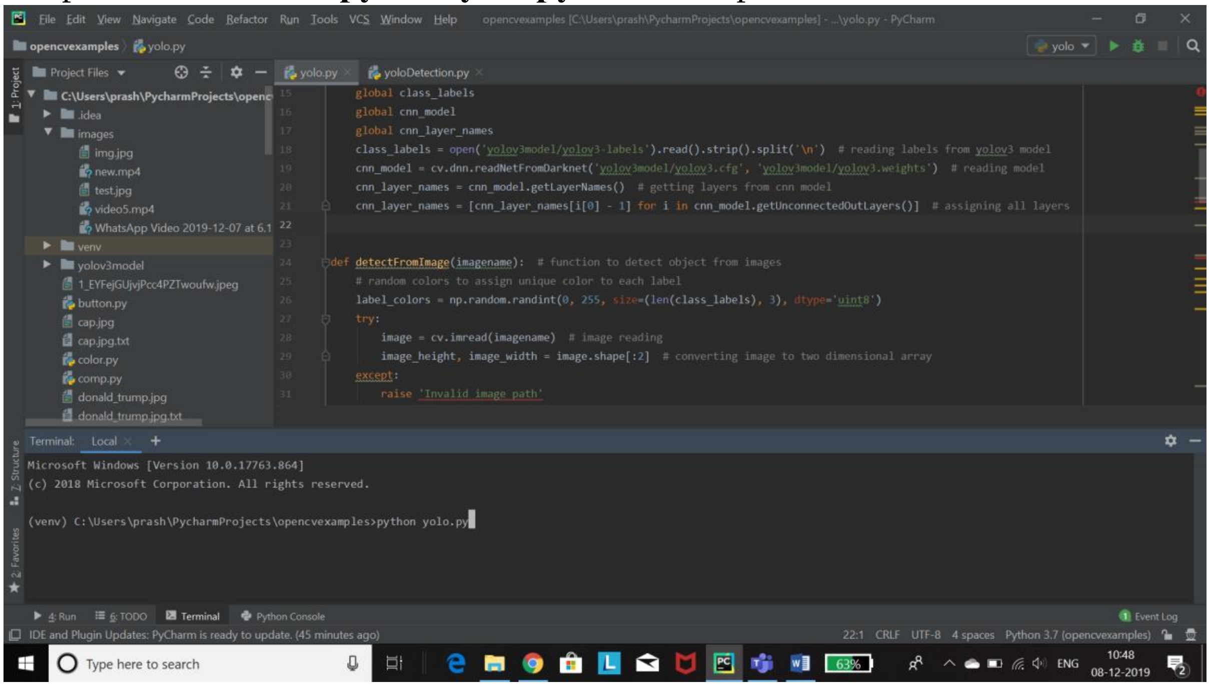
3.) In Step 3 need to go to Terminal or Command prompt as shown in the below screenshot:



```
    global class_labels
    global cnn_model
    global cnn_layer_names
    class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n')  # reading labels from yolov3 model
    cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights')  # reading model
    cnn_layer_names = cnn_model.getLayerNames()  # getting layers from cnn model
    cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in cnn_model.getUnconnectedOutLayers()]  # assigning all layers


def detectFromImage(imagename):  # function to detect object from images
    # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3), dtype='uint8')
    try:
        image = cv.imread(imagename)  # image reading
        image_height, image_width = image.shape[:2]  # converting image to two dimensional array
    except:
        raise 'Invalid image path'
```

```
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. All rights reserved.

(venv) C:\Users\prash\PycharmProjects\opencvexamples>
```

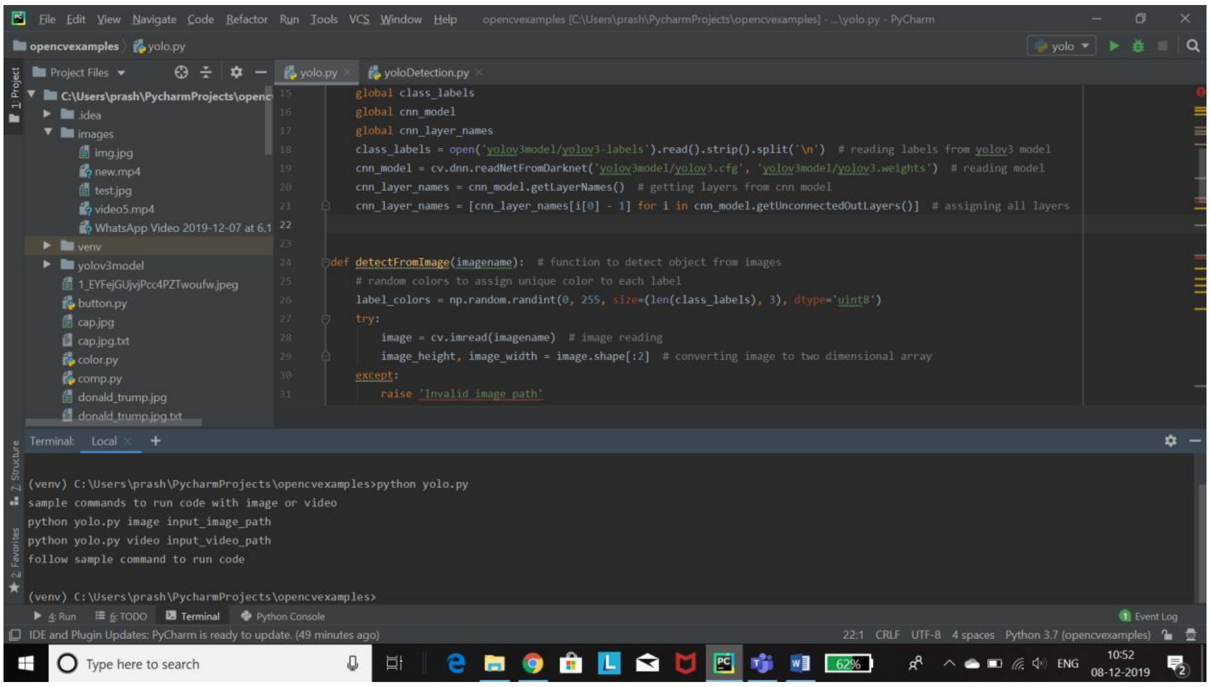
4.) In Step 4 need to enter **"python yolo.py"** after the path



```python
    global class_labels
    global cnn_model
    global cnn_layer_names
    class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n')  # reading labels from yolov3 model
    cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights')  # reading model
    cnn_layer_names = cnn_model.getLayerNames()  # getting layers from cnn model
    cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in cnn_model.getUnconnectedOutLayers()]  # assigning all layers


def detectFromImage(imagename):  # function to detect object from images
    # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3), dtype='uint8')
    try:
        image = cv.imread(imagename)  # image reading
        image_height, image_width = image.shape[:2]  # converting image to two dimensional array
    except:
        raise 'Invalid image path'
```

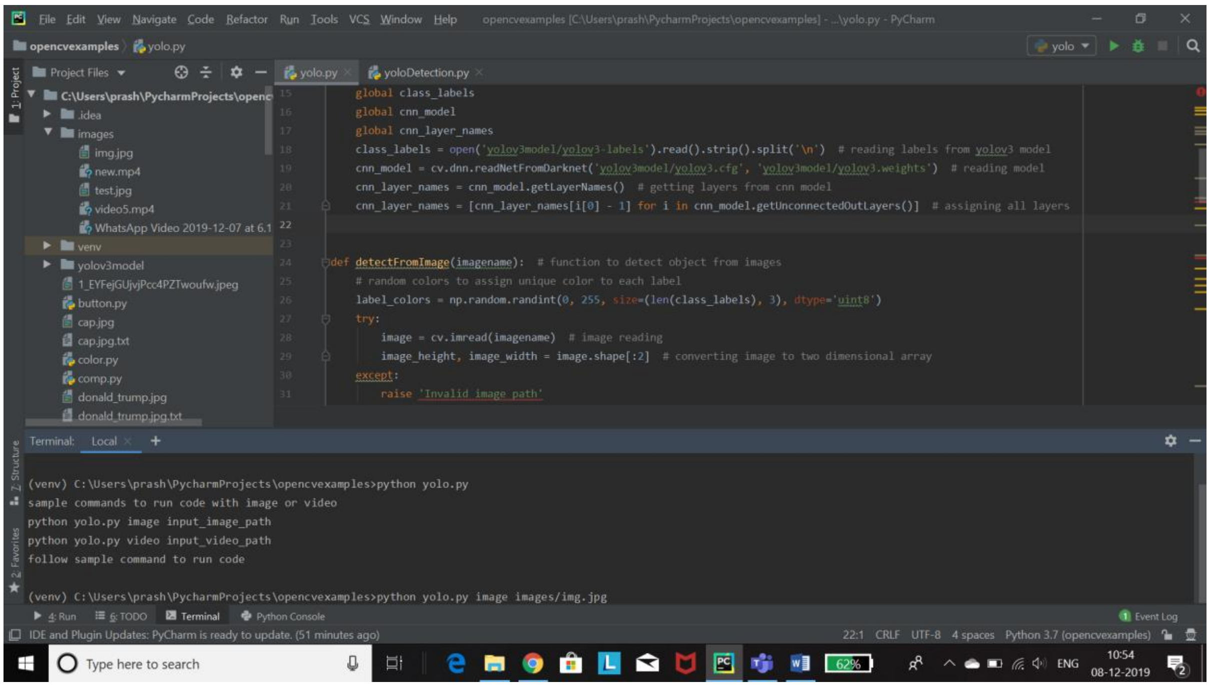
```
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. All rights reserved.

(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py
```

5.) In Step 5 after running the command, we will get two options in the terminal. One option is for identifying the objects in the image and another option for identifying the objects in the video as shown in the below screenshot



```
(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py
sample commands to run code with image or video
python yolo.py image input_image_path
python yolo.py video input_video_path
follow sample command to run code

(venv) C:\Users\prash\PycharmProjects\opencvexamples>
```

6.) In Step 6 Next we need to identify the objects present in the image, we need to pass the path of the sample images that are presented in the path. So, we give the path **"python yolo.py image images/img.jpg".** This command python is the software name and yolo.py is the program name and ‘image’ means we want application to detect object from image and ‘images/img.jpg’ is the input image. Similarly, for videos instead of image we need to pass video with video path.

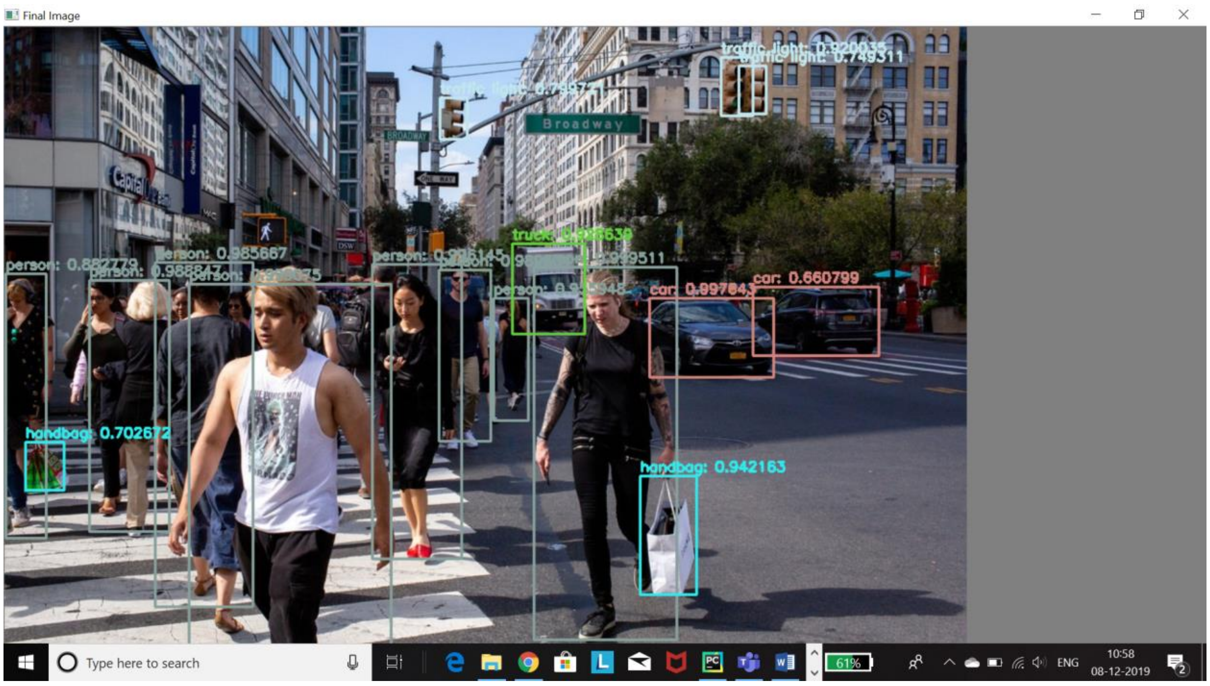


```python
    global class_labels
    global cnn_model
    global cnn_layer_names
    class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n')  # reading labels from yolov3 model
    cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights')  # reading model
    cnn_layer_names = cnn_model.getLayerNames()  # getting layers from cnn model
    cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in cnn_model.getUnconnectedOutLayers()]  # assigning all layers


def detectFromImage(imagename):  # function to detect object from images
    # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3), dtype='uint8')
    try:
        image = cv.imread(imagename)  # image reading
        image_height, image_width = image.shape[:2]  # converting image to two dimensional array
    except:
        raise 'Invalid image path'
```

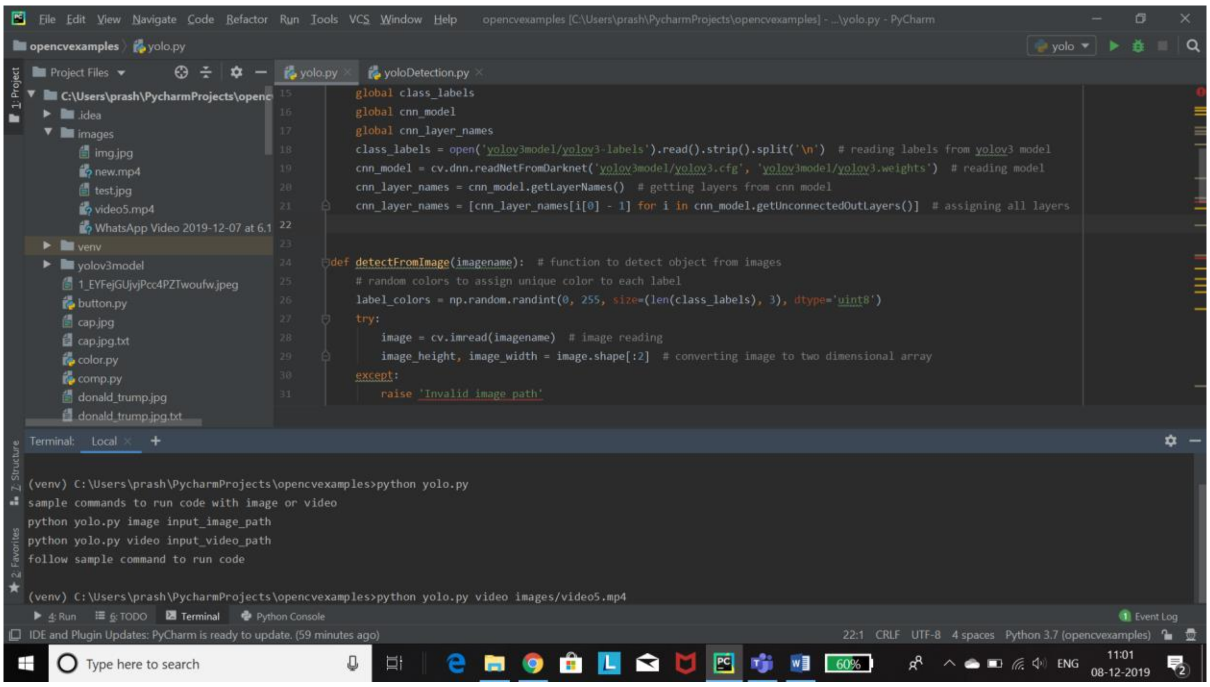
```
(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py
sample commands to run code with image or video
python yolo.py image input_image_path
python yolo.py video input_video_path
follow sample command to run code

(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py image images/img.jpg
```

opencvexamples [C:\Users\prash\PycharmProjects\opencvexamples] - ...\yolo.py - PyCharm
yolo.py
yoloDetection.py
Project Files
Terminal: Local
IDE and Plugin Updates: PyCharm is ready to update. (51 minutes ago)

7.) In step 7 we need to run the command which we have given in the step5. Then we will get the below output where all the objects in the given image will be displayed as shown in the below screenshot.



8.) In step 8 we need to identify the objects that are present in the video, So We need to pass the path of the sample video that are presented in the path. So, we give the path "**python yolo.py video images/video5.mp4**" as shown in the below screenshot. When we are giving video then it will take time to extract all frames from video for object detection and then create a new video called 'what.avi' in the same code folder. If it's taking long time, then you can press CTRL+C to stop execution and then you can play '**what.avi**' file.



```python
    global class_labels
    global cnn_model
    global cnn_layer_names
    class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n')  # reading labels from yolov3 model
    cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights')  # reading model
    cnn_layer_names = cnn_model.getLayerNames()  # getting layers from cnn model
    cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in cnn_model.getUnconnectedOutLayers()]  # assigning all layers


def detectFromImage(imagename):  # function to detect object from images
    # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3), dtype='uint8')
    try:
        image = cv.imread(imagename)  # image reading
        image_height, image_width = image.shape[:2]  # converting image to two dimensional array
    except:
        raise 'Invalid image path'
```

```
(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py
sample commands to run code with image or video
python yolo.py image input_image_path
python yolo.py video input_video_path
follow sample command to run code

(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py video images/video5.mp4
```

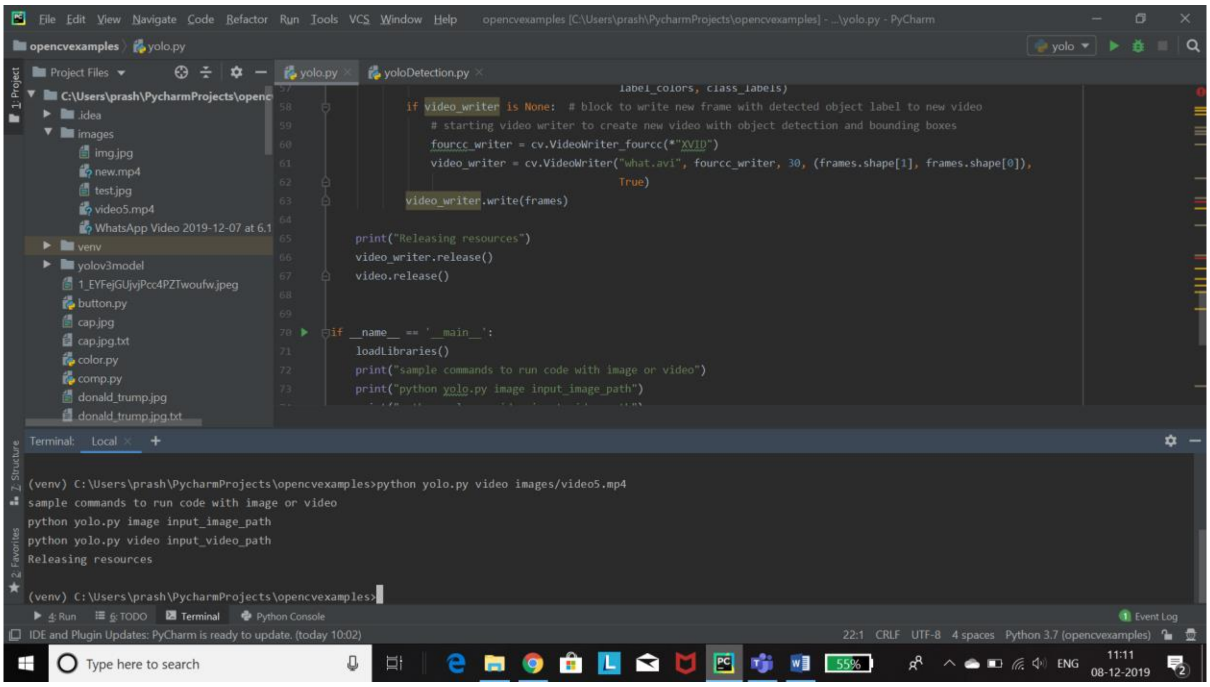
IDE and Plugin Updates: PyCharm is ready to update. (59 minutes ago)

File Edit View Navigate Code Refactor Run Tools VCS Window Help
opencvexamples [C:\Users\prash\PycharmProjects\opencvexamples] - ...\yolo.py - PyCharm
opencvexamples
yolo.py
Project Files
C:\Users\prash\PycharmProjects\opencv
.idea
images
img.jpg
new.mp4
test.jpg
video5.mp4
WhatsApp Video 2019-12-07 at 6.1
venv
yolov3model
1_EYFejGUjvjPcc4PZTwoufw.jpeg
button.py
cap.jpg
cap.jpg.txt
color.py
comp.py
donald_trump.jpg
donald_trump.jpg.txt
yoloDetection.py
Terminal: Local
4: Run
6: TODO
Terminal
Python Console
Event Log
22:1 CRLF UTF-8 4 spaces Python 3.7 (opencvexamples)
Type here to search
ENG
11:01
08-12-2019

9.) In step 9 the execution will get starts and the given input will be executed and generates the output and save the output video in the name of "**what.avi**" name in the same folder. Then we need to go the folder and we can see the output where the given video will be played by detecting the objects that are presented in it for each frame.

Note: The input size should be less than the 3 MB. It should not be more than that size.

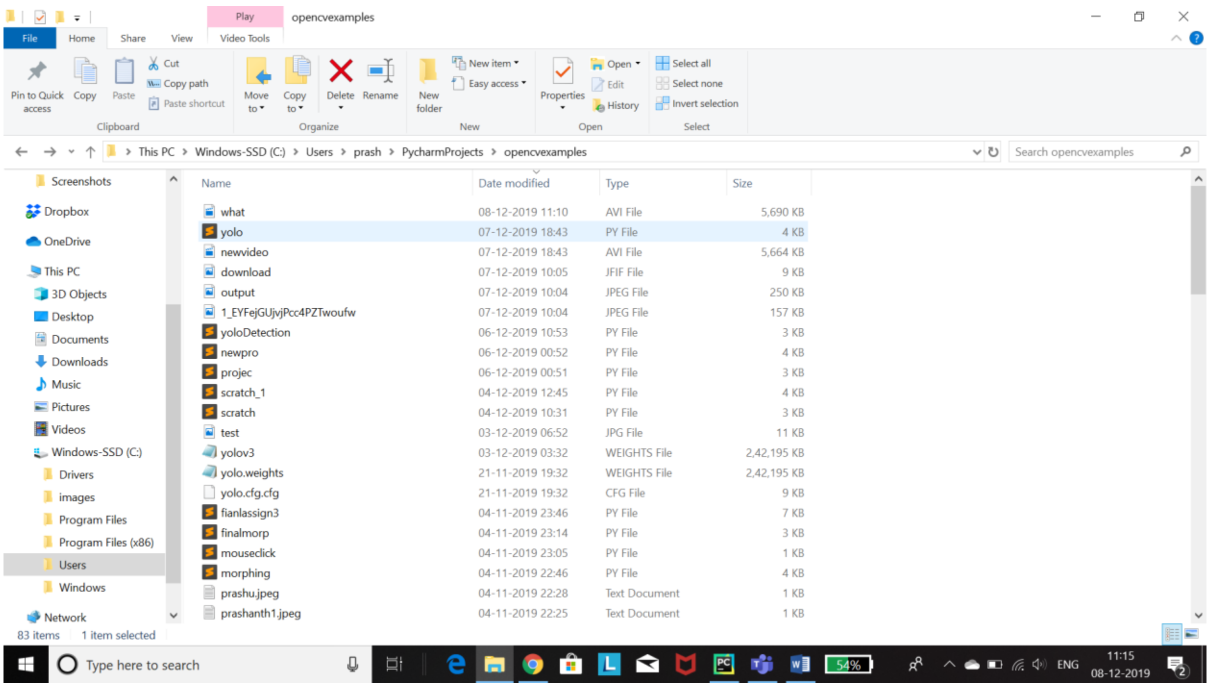
10.) In step 10 after the video execution gets finished all the resources will get released and the same message will be displayed on the terminal as shown in the below screenshot. Now the output video is ready for our view.



```python
                                        label_colors, class_labels)
            if video_writer is None:  # block to write new frame with detected object label to new video
                # starting video writer to create new video with object detection and bounding boxes
                fourcc_writer = cv.VideoWriter_fourcc(*"XVID")
                video_writer = cv.VideoWriter("what.avi", fourcc_writer, 30, (frames.shape[1], frames.shape[0]),
                                              True)
            video_writer.write(frames)

    print("Releasing resources")
    video_writer.release()
    video.release()


if __name__ == '__main__':
    loadLibraries()
    print("sample commands to run code with image or video")
    print("python yolo.py image input_image_path")
```

```
(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py video images/video5.mp4
sample commands to run code with image or video
python yolo.py image input_image_path
python yolo.py video input_video_path
Releasing resources

(venv) C:\Users\prash\PycharmProjects\opencvexamples>
```

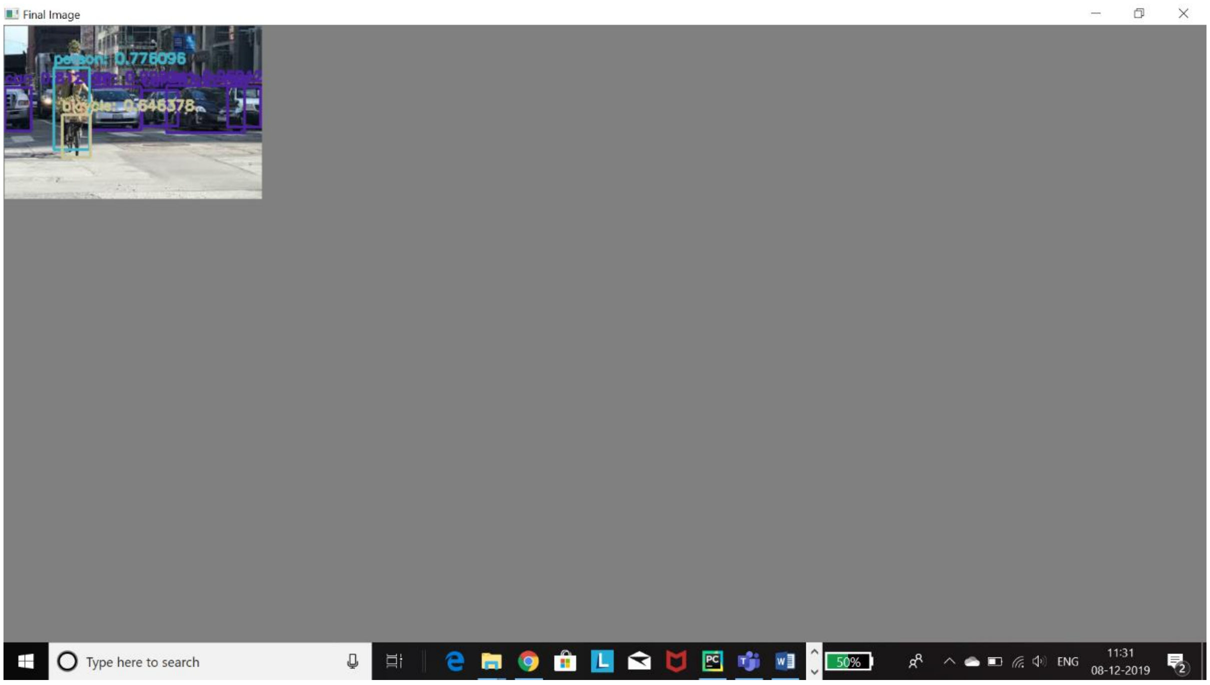
11.) In step 11 we need to go to that folder to see the output video with the name like "what.avi" we have given in the program. We must get to that path and we can see that output video.

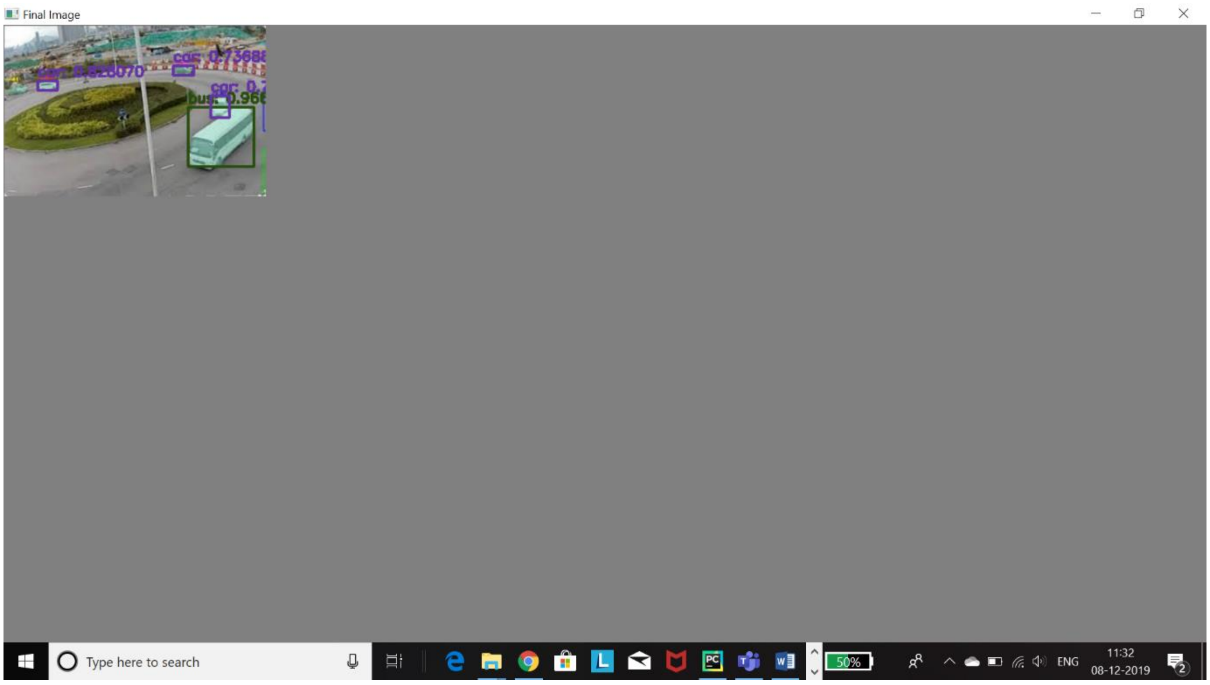

This PC > Windows-SSD (C:) > Users > prash > PycharmProjects > opencvexamples

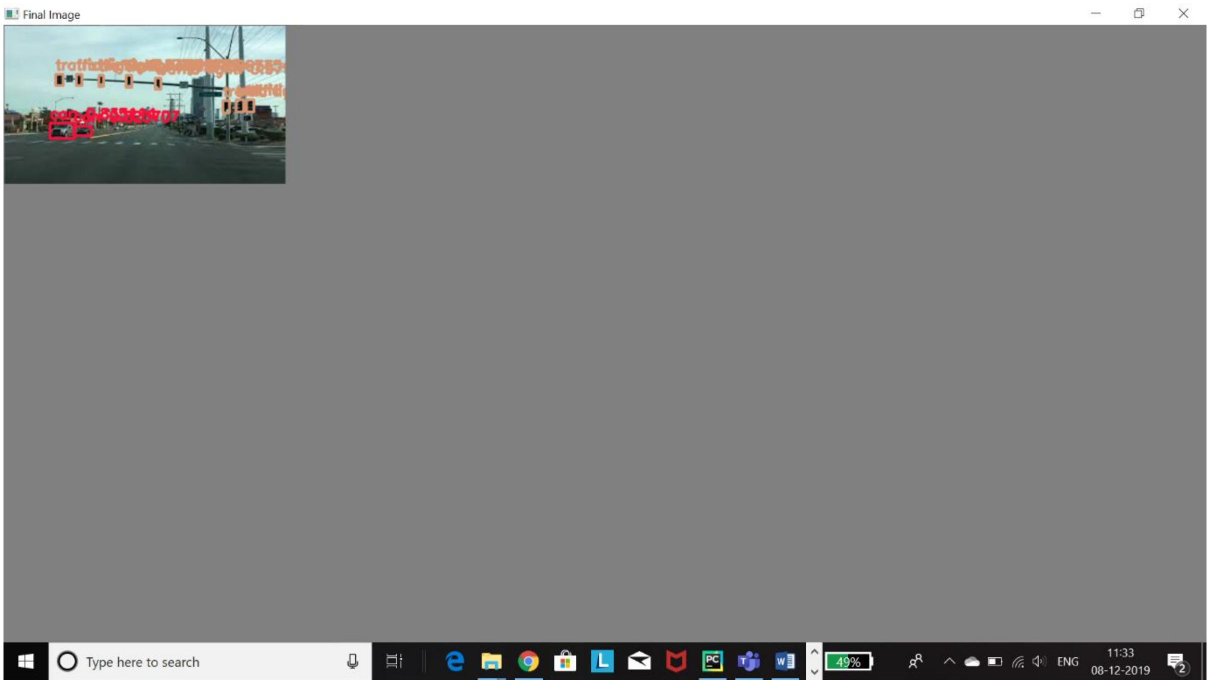
Name	Date modified	Type	Size
what	08-12-2019 11:10	AVI File	5,690 KB
yolo	07-12-2019 18:43	PY File	4 KB
newvideo	07-12-2019 18:43	AVI File	5,664 KB
download	07-12-2019 10:05	JFIF File	9 KB
output	07-12-2019 10:04	JPEG File	250 KB
1_EYFejGUjvjPcc4PZTwoufw	07-12-2019 10:04	JPEG File	157 KB
yoloDetection	06-12-2019 10:53	PY File	3 KB
newpro	06-12-2019 00:52	PY File	4 KB
projec	06-12-2019 00:51	PY File	3 KB
scratch_1	04-12-2019 12:45	PY File	4 KB
scratch	04-12-2019 10:31	PY File	3 KB
test	03-12-2019 06:52	JPG File	11 KB
yolov3	03-12-2019 03:32	WEIGHTS File	2,42,195 KB
yolo.weights	21-11-2019 19:32	WEIGHTS File	2,42,195 KB
yolo.cfg.cfg	21-11-2019 19:32	CFG File	9 KB
fianlassign3	04-11-2019 23:46	PY File	7 KB
finalmorp	04-11-2019 23:14	PY File	3 KB
mouseclick	04-11-2019 23:05	PY File	1 KB
morphing	04-11-2019 22:46	PY File	4 KB
prashu.jpeg	04-11-2019 22:28	Text Document	1 KB
prashanth1.jpeg	04-11-2019 22:25	Text Document	1 KB

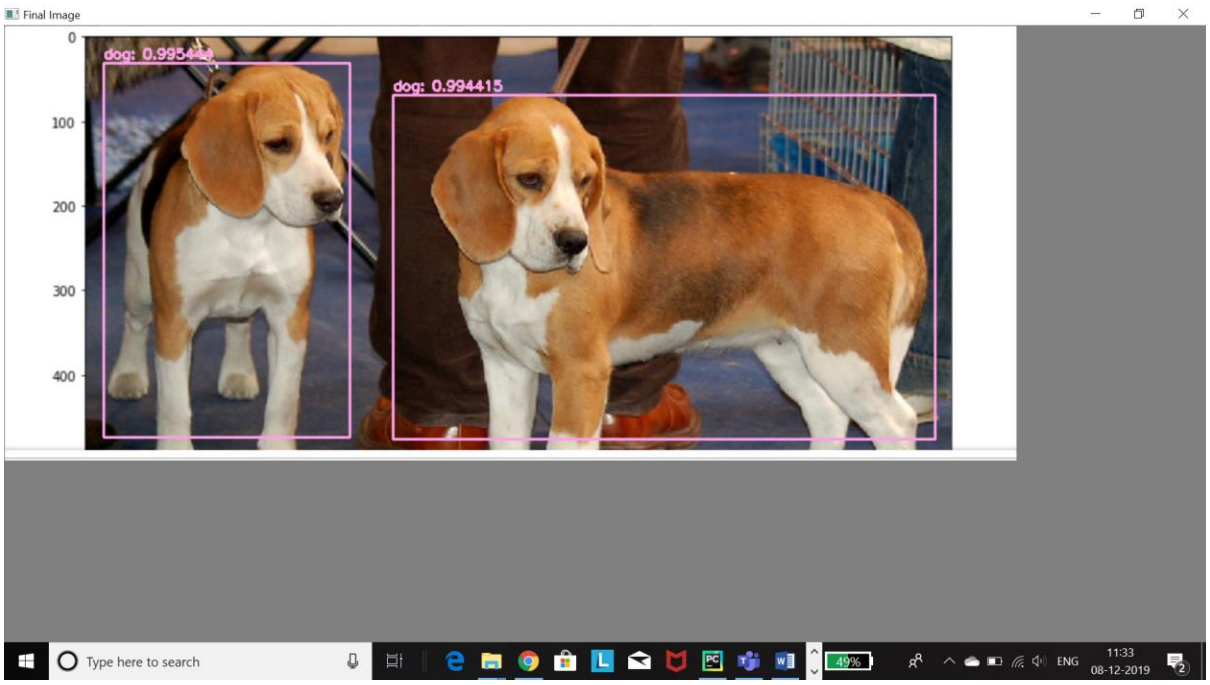
83 items 1 item selected

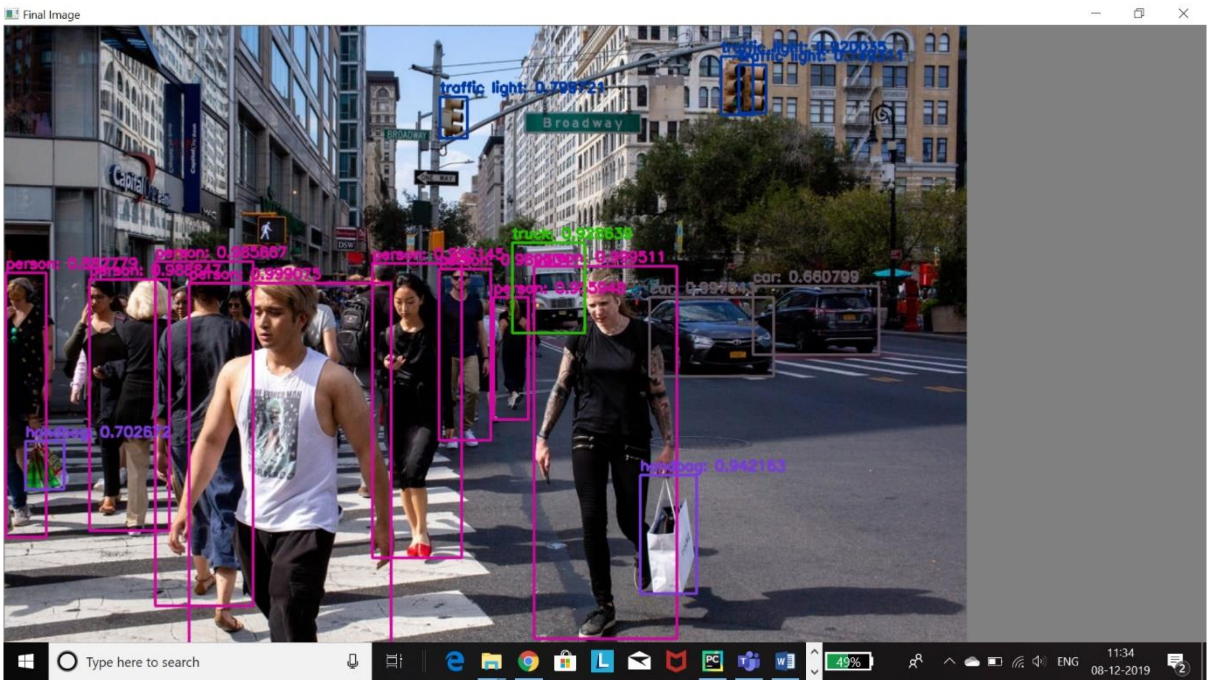
Image Outputs:

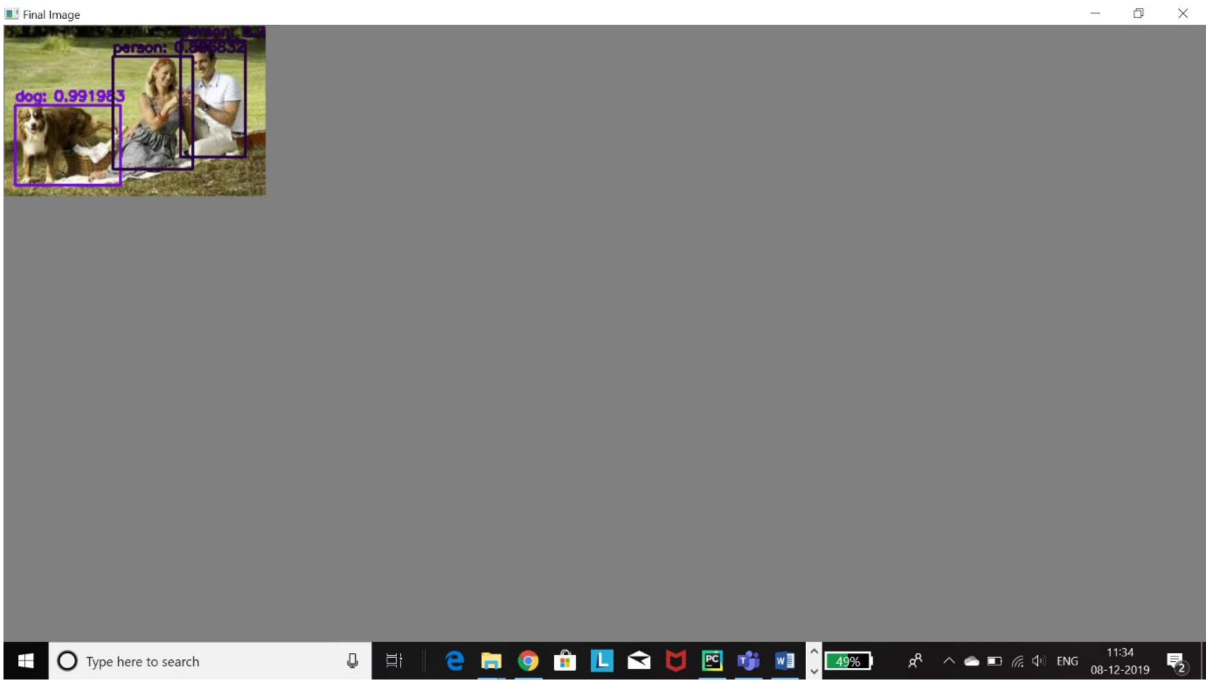



Final Image
car: 0.820070
car: 0.73688
car: 0.
bus: 0.956
Type here to search
50%
ENG
11:32
08-12-2019


Final Image
Type here to search
49%
ENG
11:33
08-12-2019


Final Image
dog: 0.995449
dog: 0.994415
0
100
200
300
400
Type here to search
49%
ENG
11:33
08-12-2019


Final Image
traffic light: 0.799721
traffic light: 0.920035
Broadway
truck: 0.926639
person: 0.985867
person: 0.988847
person: 0.999075
person: 0.999511
car: 0.660799
handbag: 0.702672
handbag: 0.942163
Type here to search
49%
ENG
11:34
08-12-2019



Functions that were used in the code for detecting objects

List of Functions used in our code to identify objects in an image

Detect Object, label bounding boxes, list bounding boxes, display image,

Load libraries, detect from image, detect from video.

1.Detect Object

An image contains multiple objects and to detect an object in an image it firstly divides an image into n*n cells and each cell has length, width and layer. Detect object function detects the object in an image based on size and aspect ratio.

2.Label Bounding Boxes

Label bounding box uses confidence value to label each object in an image. Confidence value describes how close an object is related to available features. If confidence value is high, then accuracy is high and if confidence value is low then accuracy is low. The confidence value is directly propositional to accuracy.

3.List Bounding Boxes

List bounding boxes functions characterizes labels and gives names for each label with the help of classifiers.

4.Display Image

Display image function displays the name of every object in an image. It displays names of multiple objects in an image.

5.Load libraries

Load libraries loads Global class labels, global CNN models, global CNN names.

In yoloDetection.py python file we have totally four functions namely

- detect Object()
- labelsBoundingBoxes(),
- listBoundingBoxes()
- displayImage().

  Firstly, detectObject() function is used to detect the objects in the image or in the frames that are presented in the video. Secondly, labelsBoundingBoxes() function is used to label those bounding boxes. Thirdly, listBoundingBoxes() function is used to list those bounding boxes. Lastly, displayImage() function is used to display the image in the new window. In detectObject() we use cv.dnn.blobFromImage() function which is used to facilitate the pre-processing of the image. We use CNNnet.setInput(blob_object) function that sets new input to the network. CNNnet.forward(total_layer_names) function is used to run the inference through the given network and then collects the predictions from all the total number of layers. listBoundingBoxes() function is used to list all the bounding boxes in an image with the cnn out layers, height of the image, width of the image. cv.dnn.NMSBoxes() function is used to perform the Non Maximum Suppression Boxes for eliminating the reduand with some low confidence value. Then if Boundingboxes is None or confidence_value is None or ids is None or class_ids is None is an if condition which is used to check whether the bounding boxes, confidence value, id's and class id's is loaded. labelsBoundingBoxes() function is used to label the bounding boxes that contains the image, bounding boxes, confidence value, class ids, ids, name colors, class labels. Then we return the image, bounding boxes, confidence value, class ids and ids. In labelsBoundingBoxes() we draw the boxes. cv.rectangle() is used to draw the image of the rectangle. cv.putText() is used to draw the text string for any image and finally it returns the image. listBoundingBoxes() is a function which is used to list the bounding boxes and it takes the input as image, height of the image, width of the image and the threshold confidence value. It contains the box_array, confidence_array, class_ids_array. np.argmax() function is used to returns the indices of the maximum element of the array in the axis. if confidence_value > threshold_conf: is a condition which is used to check that the confidence value is greater than the threshold value. box_array.append is used to append the xx, yy, width of the box and height of the box to the box array. confidence_array.append() is used to append the confidence value to the confidence array. class_ids_array.append() is used to append the class id to the class ids array. then it finally returns the box array, confidence array, class id array. displayImage() is used to display an image in the window where cv.imshow() function is used to display an image in the window. cv.waitkey() is used to return the user pressed key to activate the window.

In yolo.py we totally use three functions. They are loadLibraries(), detectFromImage() and detectFromVideo(). The first function is loadLibraries() function is used to load yolov3 model weight and class labels. It actually uses global class_labels, global cnn_model, global cnn_layer_names. We actually open the yolov3model/yolov3-labels which is used to read the labels from yolov3 model. cv.dnn.readNetFromDarknet() function is used to read the model from the dark net. cnn_model.getLayerNames() function is used to getting the layers from the cnn model. then we will assign all the layers. The Second function is detectFromImage() which is used to detect the objects from the image. np.random.randint() function is used where random colors to assign unique color to the each label. cv.imread() function is used in image reading. Then we convert the image to two dimensional array. detectObject() function is used to call the detection function. Then finally we display the image with detected objects label using displayImage(). The Third function is detectFromVideo() function is used to read objects from video. np.random.randint() function where the random colors to assign unique color to each label. cv.VideoCapture() function is used to print the load the videos. Then we read the video from the given path. video.read() function is done by taking each frame from the video. if not frame_grabbed: is a condition to check whether video loaded or not. Then we detect the object from the frame. if video_writer is None: is a condition block to write new frame with detected object label to new video. cv.VideoWriter_fourcc() function is used to starting video writer to create new video with object detection and bounding boxes. then we release the video.

5) Put the major part of your code and comments in the report.

YoloDetection.py

```
import numpy as np
import argparse
import cv2 as cv
import subprocess
import time
import os


# function to detect the object
def detectObject(CNNnet, total_layer_names, image_height, image_width,
image, name_colors, class_labels,
            Boundingboxes=None, confidence_value=None, class_ids=None,
ids=None, detect=True):
    if detect:

        # blobFromImage function is used to facilitate the image pre processing
        blob_object = cv.dnn.blobFromImage(image, 1 / 255.0, (416, 416),
swapRB=True, crop=False)

        # function that sets new input to the network
        CNNnet.setInput(blob_object)

        # it is used to run the inference through the given network and collects the
predictions from all the total layers
        cnn_outs_layer = CNNnet.forward(total_layer_names)

        # it is used to list the bounding boxes in an image
        Boundingboxes, confidence_value, class_ids =
listBoundingBoxes(cnn_outs_layer, image_height, image_width, 0.5)

        # It performs the NonMaximumSuppression for eliminating the reduand
with some low confidence value

        ids = cv.dnn.NMSBoxes(Boundingboxes, confidence_value, 0.5, 0.3)

        # condition to check whether the bounding boxes, confidence value, ids
and class ids is loaded
        if Boundingboxes is None or confidence_value is None or ids is None or
class_ids is None:
            raise '[ERROR] unable to draw boxes.'

        # function to label the bounding boxes
        image = labelsBoundingBoxes(image, Boundingboxes, confidence_value,
class_ids, ids, name_colors, class_labels)
```

```
    # returns the image, boundingboxes, confidence values, class ids and ids.
    return image, Boundingboxes, confidence_value, class_ids, ids


# function to label the bouding boxes
def labelsBoundingBoxes(image, Boundingbox, conf_thr, classID, ids,
color_names, predicted_labels):
    if len(ids) > 0:
        for i in ids.flatten():

            # draw boxes
            xx, yy = Boundingbox[i][0], Boundingbox[i][1]
            width, height = Boundingbox[i][2], Boundingbox[i][3]

            class_color = [int(color) for color in color_names[classID[i]]]

            # it is used to draw the image rectangle
            cv.rectangle(image, (xx, yy), (xx + width, yy + height), class_color, 2)
            text_label = "{}: {:4f}".format(predicted_labels[classID[i]], conf_thr[i])
            # it is used to draw the text string for any image

            cv.putText(image, text_label, (xx, yy - 5),
cv.FONT_HERSHEY_SIMPLEX, 0.5, class_color, 2)

    # it returns the image
return image

# function to list the bouding boxes
def listBoundingBoxes(image, image_height, image_width, threshold_conf):
    box_array = []
    confidence_array = []
    class_ids_array = []

    for img in image:
        for obj_detection in img:
            detection_scores = obj_detection[5:]

  # it returns the indices of the maximum element of the array in axis
            class_id = np.argmax(detection_scores)
            confidence_value = detection_scores[class_id]
```

```
 # condition to check that the confidence value is greater than the threshold
value
            if confidence_value > threshold_conf:
                Boundbox = obj_detection[0:4] * np.array([image_width,
image_height, image_width, image_height])
                center_X, center_Y, box_width, box_height = Boundbox.astype('int')

                xx = int(center_X - (box_width / 2))
                yy = int(center_Y - (box_height / 2))

   # to append the xx, yy, width of the box and height of the box to the box array
                box_array.append([xx, yy, int(box_width), int(box_height)])

 # to append the confidence value to the confidence array
                confidence_array.append(float(confidence_value))

 # to append the class id to the class ids array
                class_ids_array.append(class_id)

  # it returns the box array, confidence array and class id array
    return box_array, confidence_array, class_ids_array

# function to display an image in the window
def display Image(image):

  # to display an image in the window
    cv.imshow("Final Image", image)

  # to return the user pressed key to activate the window
    cv.waitKey(0)
```

YOLO.Py

```
import numpy as np
import cv2 as cv
import subprocess
import time
import os
from yoloDetection import detectObject, displayImage
import sys

global class_labels
global cnn_model
global cnn_layer_names

# function to load yolov3 model weight and class labels
def loadLibraries():
    global class_labels
    global cnn_model
    global cnn_layer_names

  # reading labels from yolov3 model
    class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n')

  # reading model from Dark net
    cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg',
'yolov3model/yolov3.weights')

 # getting layers from cnn model
    cnn_layer_names = cnn_model.getLayerNames()

    # assigning all layers
    cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in
cnn_model.getUnconnectedOutLayers()]


# function to detect object from images
def detectFromImage(imagename):

  # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3),
dtype='uint8')
    try:

  # image reading
        image = cv.imread(imagename)

    # converting image to two dimensional array
        image_height, image_width = image.shape[:2]
    except:
        raise 'Invalid image path'
    finally:
        # calling detection function
        image, _, _, _, _ = detectObject(cnn_model, cnn_layer_names,
image_height, image_width, image, label_colors,
                                 class_labels)
        # display image with detected objects label
        displayImage(image)

# function to read objects from video
def detectFromVideo(videoFile):

    # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3),
dtype='uint8')
    try:

        video = cv.VideoCapture(videoFile)

 # reading video from given path
        frame_height, frame_width = None, None
        video_writer = None
    except:
        raise 'Unable to load video'
    finally:
        while True:

   # taking each frame from video
            frame_grabbed, frames = video.read()

  # condition to check whether video loaded or not
            if not frame_grabbed:
                break
            if frame_width is None or frame_height is None:

 # detecting object from frame
                frame_height, frame_width = frames.shape[:2]
            frames, _, _, _, _ = detectObject(cnn_model, cnn_layer_names,
frame_height, frame_width, frames,
                                       label_colors, class_labels)

  # block to write new frame with detected object label to new
video
            if video_writer is None:

  # starting video writer to create new video with object detection and bounding
boxes
                fourcc_writer = cv.VideoWriter_fourcc(*"XVID")
            video_writer = cv.VideoWriter("what.avi", fourcc_writer, 30,
(frames.shape[1], frames.shape[0]),
                                True)
        video_writer.write(frames)

    print("Releasing resources")
    video_writer.release()
    video.release()


if __name__ == '__main__':
    loadLibraries()
    print("sample commands to run code with image or video")
    print("python yolo.py image input_image_path")
    print("python yolo.py video input_video_path")
    if len(sys.argv) == 3:
        if sys.argv[1] == 'image':
            detectFromImage(sys.argv[2])
        elif sys.argv[1] == 'video':
            detectFromVideo(sys.argv[2])
        else:
            print("invalid input")
    else:
        print("follow sample command to run code")

# video_path = None
# video_output_path = "out.avi"
```

Introduction:

General problems that occour:

The main problem that comes with the object detection is that identifying of the object in the particular video or an audio. Accuracy also plays a key and crucial role because it defines the standard of the algorithm. Compared to older techniques the deep learning and other methods came into existence and helped to increase the accuracy percentage of the image or an video. These were advanced techniques in use right now. The main goal of object detection is to predict or identify the classes in a particular image. Classes are nothing but types. Ex- cat, dog, ball, book are examples of class. The algorithm should firstly do the clasification always . After identifying the class the algorithm should create the bounding boxes around that class. This process is called as localization. This plays a key role. If both these things combine together they get complicated and that we have to solve. This process is called as object detection. The image or video is supplied to the system as input and the final result would be a combination of classification and localization with labels.

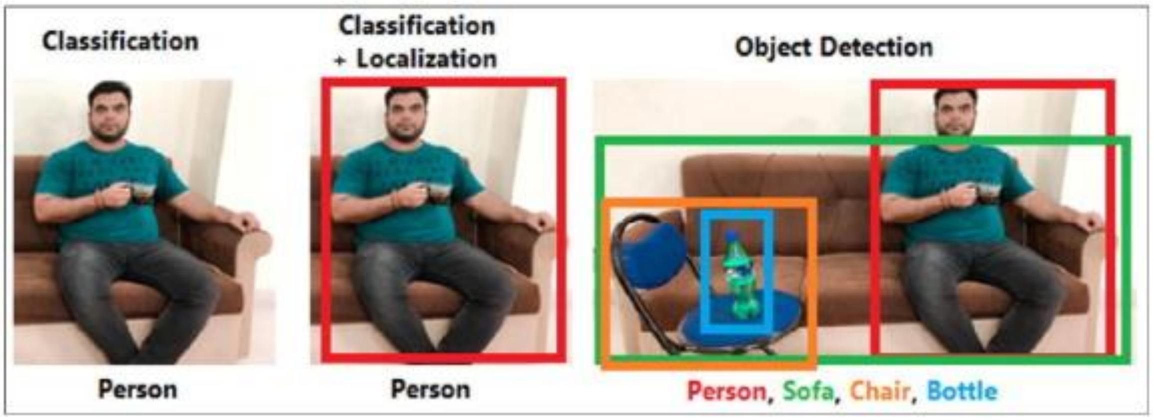


Fig1: Example of classification, localization and object detection

Applications of object detection:

There are various aspects in which we do the object detection and wide variety of uses are there. Various fields requires this and some of them are

- Automatic vehicles need to be trained with objects
- Face recognition
- Face expression recognition
- Robotic applications
- Medical use
- Traffic monitoring
- Government purposes
- Security Related identifying purposes and lot more

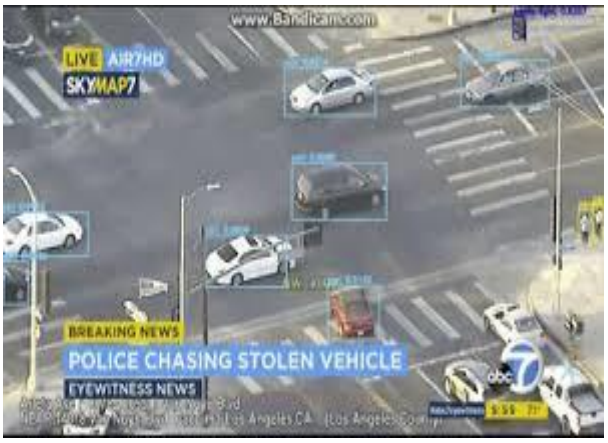


FIG: Police chasing stolen vehicles

Similar works:

Lot of research works and many plans have been executed for the concept of objection. Sliding windows were used in the olden days to detect objects. New techniques were generated apart from the old one. Some old methods used deformable part models. Deep learning methods have high accuracy and these models had the less accuracy. Some of the latest deep learning techniques are

- Convolutional neural network(CNN)
- Faster RCNN
- RCNN
- Fast RCNN
- Unified YOLO
- SSD

The things which are used in the project are depicted.

Creation and generation of bounding boxes:

Bounding boxes come under the concept of localization. These are in the shape of a square or rectangle and has a particular color to get illuminated. It is used to fit a particular part of the image into it. They come into existence for every instance and object of the image. It has particular shapes and dimensions . there are some terminology called as center-x, center-y, width and height. These are used to calculate the size of the boxes. The system need to be trained to generate them by using the predicted and the actual image size. Various algorithms are used to train them for good accuracy. Jaccard distance is the concept used to draw these boxes. It calculates between predicted and orginal by intersecting them.

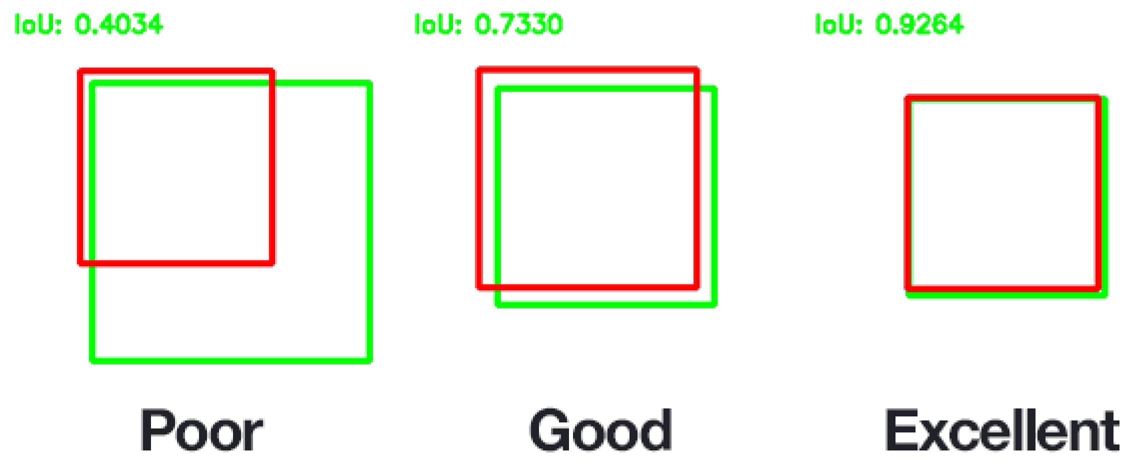


Fig: Jaccard distance measurement

Regression:

The term regression plays a key role in object detection. The main aim of this is to generate the bounding boxes around the image. All the prediction work is done by regression and then classification is used to predict what's inside the bounding box.

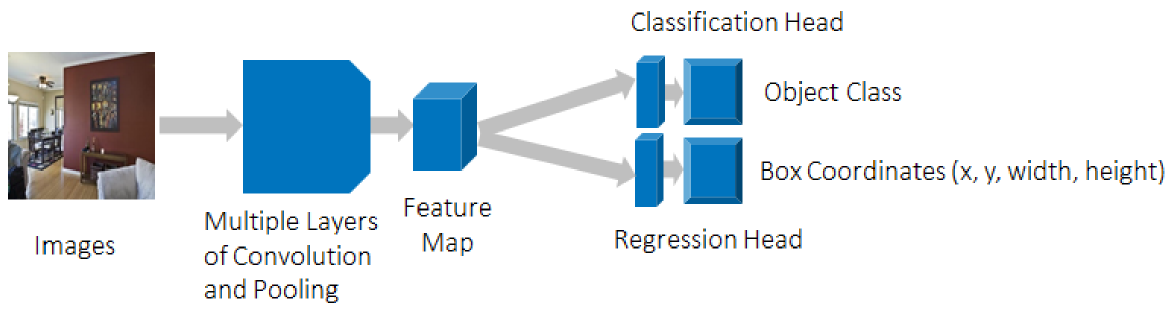

Classification Head
Object Class
Box Coordinates (x, y, width, height)
Regression Head
Images
Multiple Layers of Convolution and Pooling
Feature Map

Fig: Classification and regression

YOLO Algorithm:

Yolo is one of the latest and fast algorithm. It stands for you look only once and plays a key role in object detection. Speed of Yolo is 155 frames per second. It processes that many frames in real time. It is an open source and anyone can use that. The object representation which is generalized can be understood by the network. FCNN and YOLO work the same. Some algorithms are worked regionally like how fast RCNN does. They predict multiple regions at a time. YOLO will send image as input to FCNN with resulting output as mxm. Then the bounding boxes are generated. YOLO is so much apart from other algorithms . It considers the entire image at a time and predicts the classes and does the classification. Sliding window and particular part of image processing are all beaten by yolo. The errors which occur because of this is very less compared to other algorithms. Yolo has reduced the concept of pipeline of steps into a single and easy step. Many bounding boxes will be generated to each cell in an image. But we want only one . In order to solve the problem there will be a predictor whose primary responsibility is to predict the exact one. Non max suppression is used to predect the perfect bounding boxes.

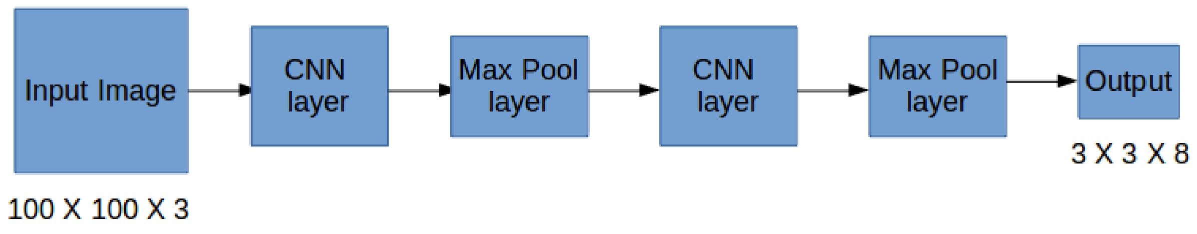
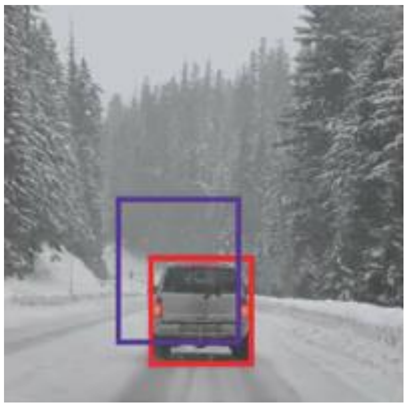


Fig: Process that goes through

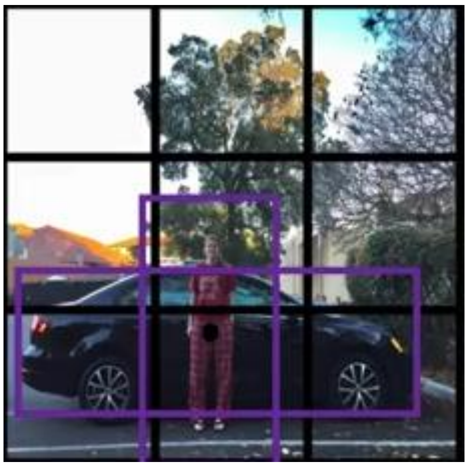
Intersection over union is the concept that helps in wether the predicted box has given us the good output or not. It is like a quality checker for the for Yolo.



In order to decide which is true wether the blue or red we use the above concept. IOU is calculated by area of intersection/ area of union. If IOU value is higher that 0.5 the prediction is true.

Anchor boxes:

Anchor boxes is the one used to detect multiple objects in a single grid. Consider a image with nxn grids and a grid has two objects in it . Then anchor box helps us in solving that.



For the above image the center point is same for all. The anchor boxes shapes are defined first itself. We can take how mnay boxes ever we want. Since the shape is same and depending on y values the shape of the girl is assigned to anchor box 1 nd the car gets assigned to anchor box 2.

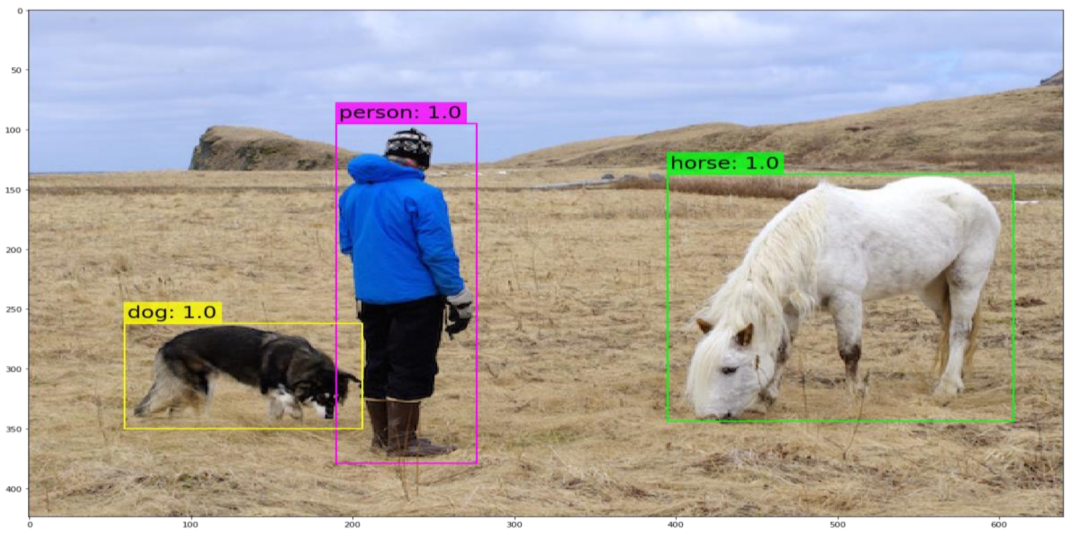


Fig: Multiple object detection by Yolo

Convolutional Neural Network(CNN)

It is one of the latest algorithm related to deep learning. It gives priority to various parts of the image. It generates weights and labels according to their priority. It differentiates the one from other. The works which are to be done before are very less compared to other methods. Filters are used in many methods and CNN uses wide variety of filters. They have the ability to learn the characters of the filters. Spatial and temporal dependencies in an image through relevant features. It analyses visually. It is also called as space invariant artificial neural networks. There will be neurons and they are interconnected to each other. It is similar to combination of biology and maths. Humans have the capability of identifying something that course in an image and we have the ability to differentiate the things in an easy and quick way in quick glance. We analyse the characters of it generate labels to it and we get full detailed description. In CNN when we send an image as input it goes through several layers . It goes through convolution, non linear , pooling, fully connected layers. Understanding the layers in an CNN is important. Based on the array dimensions the output is classified for a particular image.

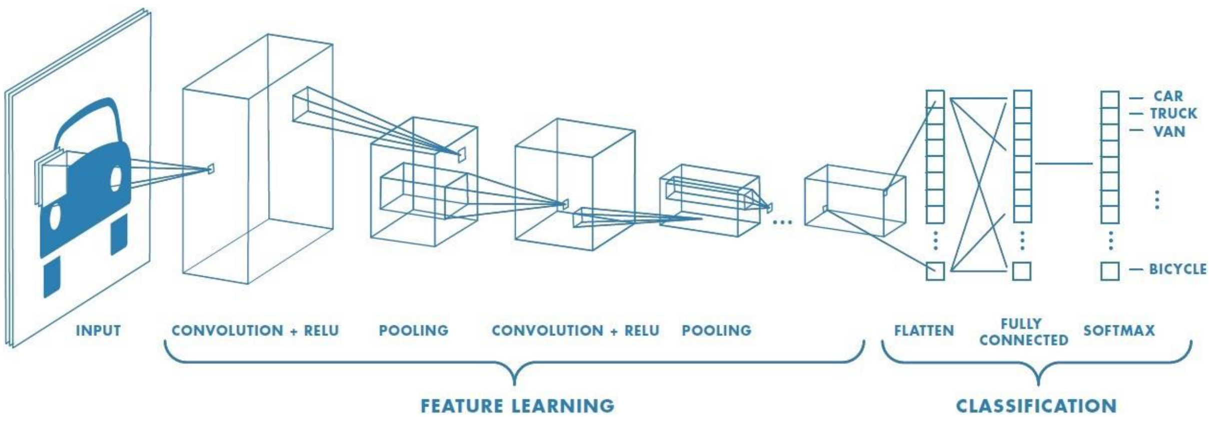
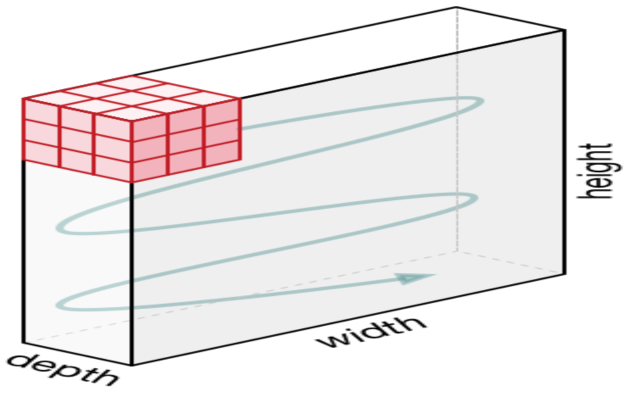


Fig: Convolutional neural network

The gap between humans and computers has been reduced y artificial intelligence. Computer vision is the field where numerous research have been undergone and lot of scope to develop. The algorithms takes input and gives priority to each blocks or parts of the picture. The pre processing which has to be done is at a very less rate compared to other algo. Respected fields play a key role and that particular area neurons will react as per the area. Images are divided based on their color space and divided based on RGB values. It splits the image in such a way that the main features of the image is not gone and the crucial part is handled safe and each sublayer is processed to extract features. The CNN divides the image in the form of multiple grids with different sizes and performs operations on that and helps in extracting the features to a output. The kernel plays a key role in performing the sub operations to the the kernel is the one which conducts the preliminary operations on the convolutional network.



This is how the kernel moves while performing the CNN

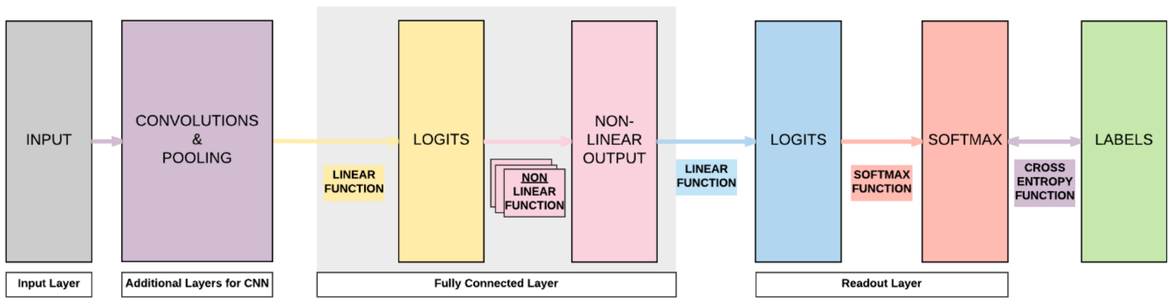


Fig: Steps that occur in CNN

The entire project has been divided in to three modules and they are listed as below.

Training Data:

Generally, we use convolutional layer for pretraining our data set. We will be using the primary 20 layers of convolutional layer for pretraining the data. We trained our network and achieved a top accuracy of 88 percentage on the ImageNet2012 in a single crop which is a very good one compared to the Caffe's Model Zoo that belongs to the Google Net models.

Design:

This current model has been implemented based on the convolutional neural network and will be evaluating using the PASCAL VOC detection dataset. The Network contains some primary convolutional layers which extracts the complete features from the given image where the layers that are fully connected will predict all the possible probabilities of the output and coordinates those outputs. The Architecture of our network got inspired from the Google Net Models for the purpose of classification of an image. The Network has totally 24 convolutional layers and followed by 2 completely connected layers. So instead of using the models proposed by the Google Net Models we will be using the 1_1 reduction layer then followed by the 3_3 convolutional layers.

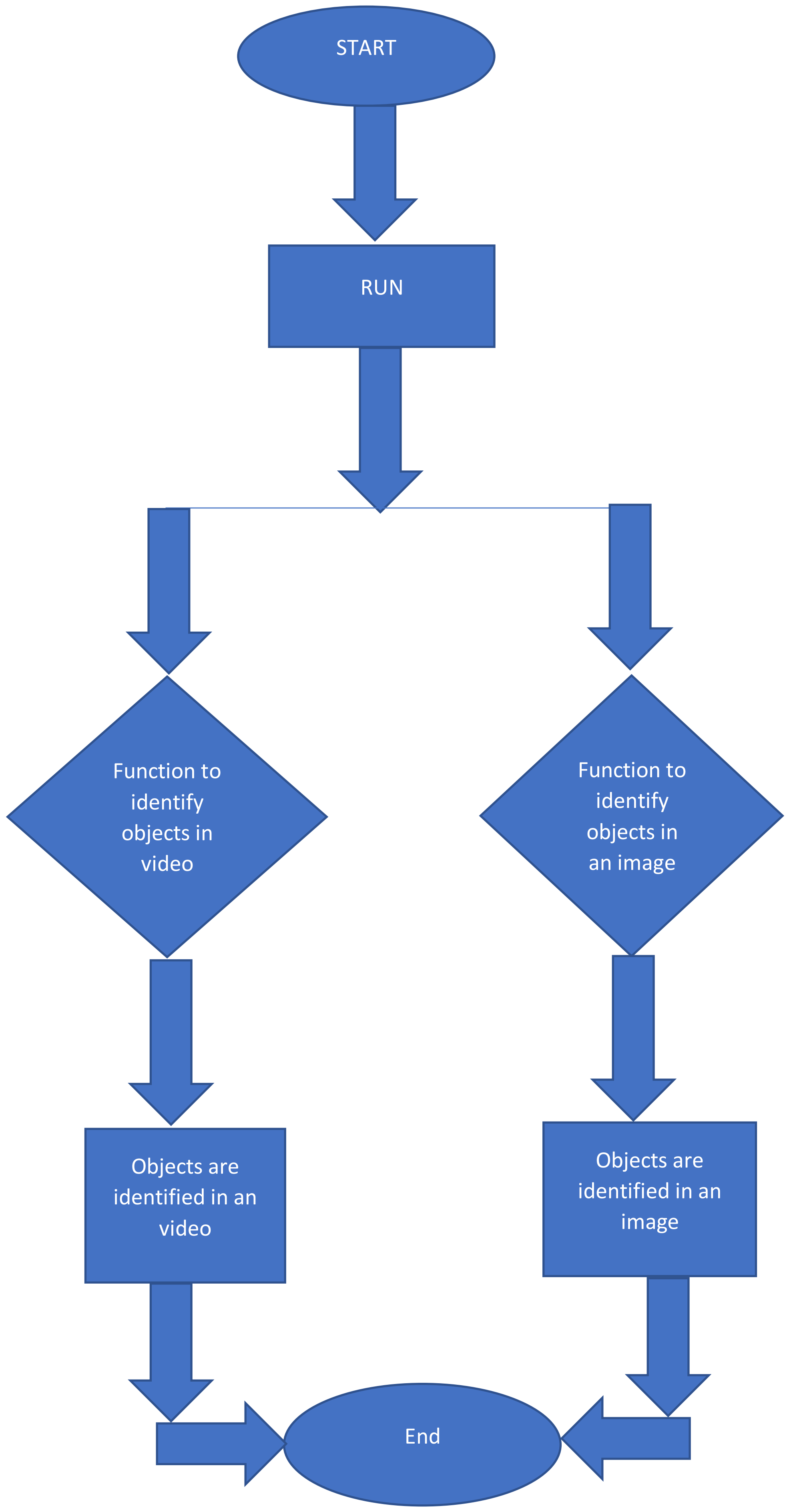
System Test:

The main purpose of doing the System testing was to identify the errors or bugs. Testing can be defined as the process where we will be trying to identify each weakness or problem that may occur in a product during the work. Testing generally, helps us to check the component, sub-components, assembled components or finished product functionality. By using the System Testing we can ensure that all the software products are meeting their standards and requirements so that it will accept all the users test cases and helps in accepting the software products with out any defects. There are different number of tests. Every test has its own requirement.

Articles are in the picture, where they will be, and how they will collaborate with each other. The human visual framework is very quick and précised one, enabling us to perform the more complex and difficult tasks like driving a car with a minimum knowledge in it. Quick, precise and accurate calculations for object discovery will enable the PCs to drive the Automobile vehicles without sensors, empower assistive gadgets to pass the constant data to humans, and to open the potentiality for broadly useful, responsive automated frameworks. Current recognition frameworks classifiers in order to perform the identification of the articles. To distinguish an article, these frameworks resizes the picture, then run the convolutional arrangement and then the Non-max concealment.

In YOLO Detection System. Handle the pictures with YOLO is a very basic and direct one. Our framework resizes the information of the picture and runs the entire convolutional model. So, that it can organize the picture and limits the subsequent identifications by the model's certainty. classifies the image and examines it at different number of areas and scales in the given test picture. Frameworks like deformable parts models (DPM) generally uses a sliding window approach where the classifier is run at equal dispersed areas over the entire picture. New approaches like R-CNN use district proposition techniques. Initially in order to create the potential to the bounding boxes in the given picture and to run a classifier on these proposed bounding boxes. After that, present handling is utilized on these refined bounding boxes, dispose the copy of these discoveries, and restoring the container dependent on the number of different types of items in the given video frame. The pipelines used here are so moderate and difficult to enhance because every individual segment had been prepared independently. We reframe the object discovery as a solitary elapse issue, directly from the picture pixels to bounding boxes and organizes them in the class probabilities. Utilizing our framework, you only look once (YOLO) at a picture to identify which items are available and where they are existed. YOLO is refreshingly straight forward one. A solitary convolutional model arrange at the same time predicts various bounding boxes and class probabilities for those crates. YOLO prepares on full pictures and legitimately streamlines identification execution. This bound together model has a few advantages for object identification. To begin with YOLO, it is a very quick one. Since we outline recognition as a elapse issue, we don't even bother with the mind-boggling pipeline. We essentially run our neural system on any other picture at test time to see the new discoveries. Our Model runs at 45 fps with no bunch handling on a Titan X GPU and a quick form runs at in excess of 150 fps. This implies that we can process spilling video progressively with under 25 milliseconds of idleness. Besides, YOLO accomplishes more than double the mean normal exactness of other constant frameworks. Secondly, YOLO reasons all around about the picture when making forecasts. Not at all like sliding window and district proposition-based methods, YOLO sees the whole picture during preparing and test time and it encodes the logical data about classes as per their appearance and look. Quick R-CNN, a top location strategy fixes in a picture for objects since it can not see the bigger setting. YOLO makes not exactly a large portion of the quantity of foundation mistakes contrasted with Fast R-CNN. Third, YOLO learns generalizable representations of articles. At the point when prepared on regular pictures and tried on fine art, YOLO outflanks top discovery strategies like DPM and R-CNN by a wide edge. Since YOLO is exceptionally generalizable it is less inclined to separate when applied to new spaces or surprising information. The entirety of our preparation and testing code is open source and can be accessible.

BLOCK DIAGRAM

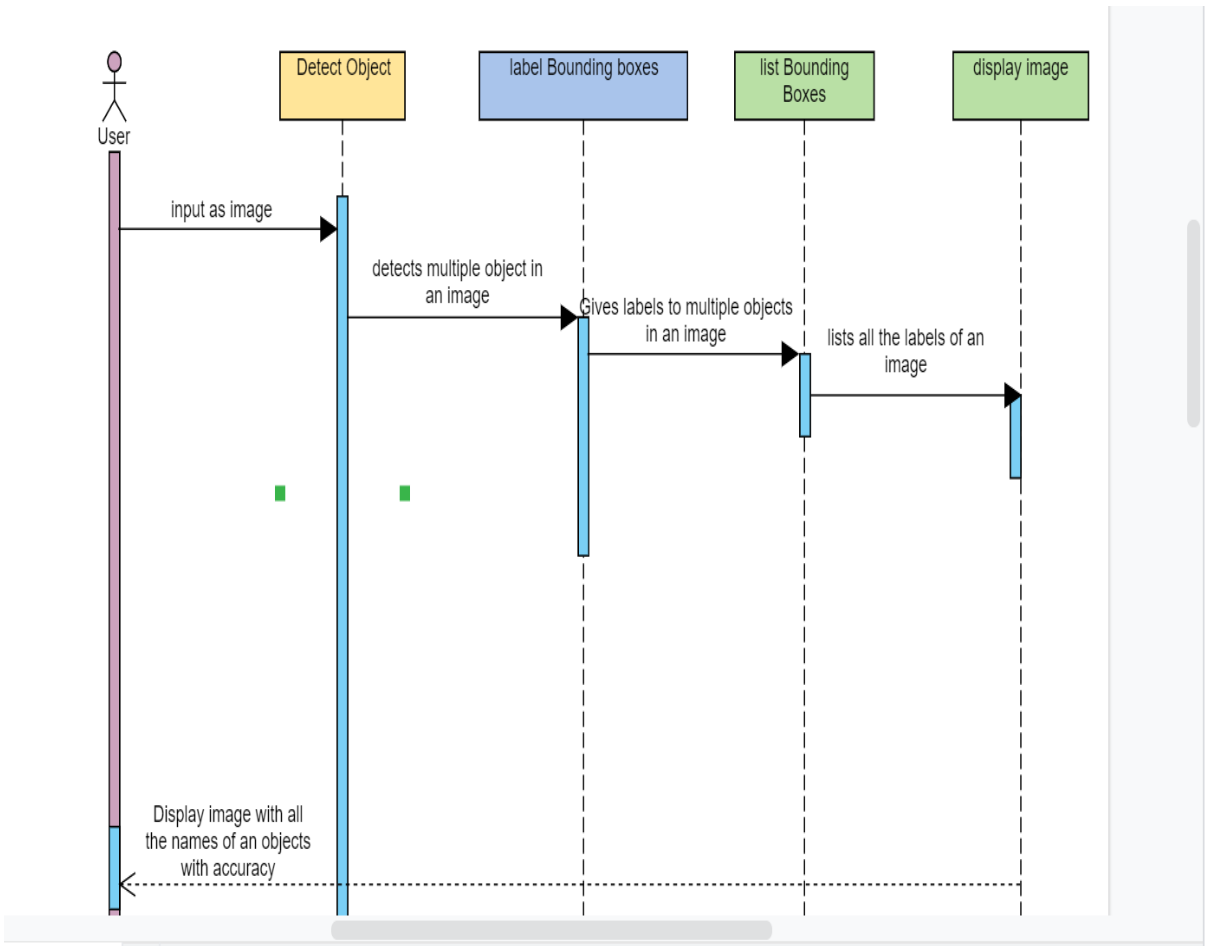

START
RUN
Function to identify objects in video
Function to identify objects in an image
Objects are identified in an video
Objects are identified in an image
End

In above block diagram, the process begins by entering an image as input. Firstly, we are giving an image or video as an input. Next it performs informative region selection, feature extraction and classification. In informative region selection, as different objects may occur at different positions and aspect ratios or sizes are different for different objects in an image. The multi-scale sliding window is used to scan a whole image. This strategy which we are using is very exhaustive and can find out all the object positions which are possible. The main problem here is it produces large number of candidate windows and some of them redundant. It takes huge amount of computer power, so it is computationally expensive. To solve this problem, we are producing only a fixed number of sliding window templates.

In feature extraction, we need extract some visual features which provides some representations which are robust and semantic in nature. SIFT (Scale Invariant feature transform), HOG (Histogram of oriented gradients) are representative features. Here we extract the features of an object in an image using different scaling techniques. It extracts all kind of features of an object. The diversity of appearances, illumination conditions, backgrounds it is little bit hard to design an feature descriptor which can perfectly describes every kind of object.

In classification, to distinguish a target object from all other different object categories we need a classifier. Classifier helps in making visual recognition more hierarchical, semantic and informative. Some of the good choices are supported vector machine (SVM), Adaboost and Deformable part-based model (DPM). With help of classifier we distinguish which feature belongs to which category.

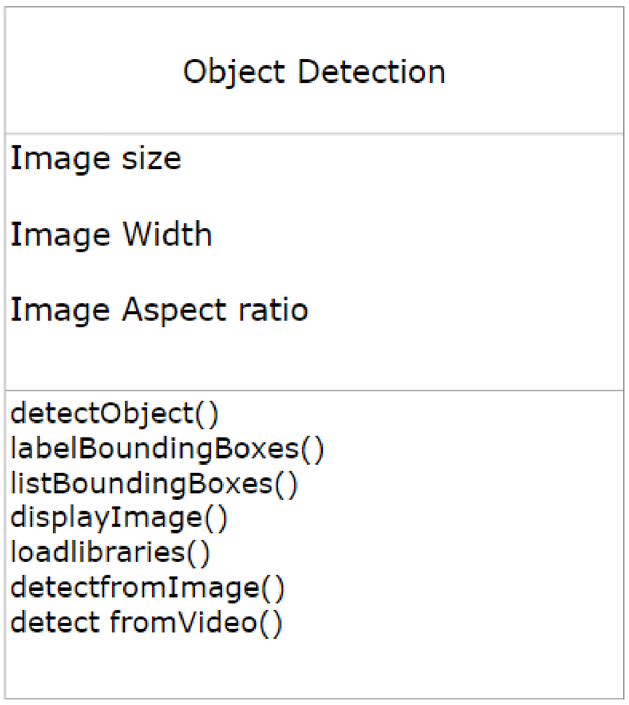
SEQENCE DIAGRAM



User
Detect Object
label Bounding boxes
list Bounding Boxes
display image
input as image
detects multiple object in an image
Gives labels to multiple objects in an image
lists all the labels of an image
Display image with all the names of an objects with accuracy

Sequence diagram describes how interactions between objects arranged in sequence of time. It explains how sequence of actions performed between the objects need to complete a task. Here firstly, user gives input as image to object detection model then it first scans the input image and then divide the image in n*n cells. By considering cell length, width it labels multiple objects in an image and list all the label objects. Finally, it display the image to user with multiple object names along with accuracy.

CLASS DIAGRAM



Class diagram describes a structure of a diagram. Static picture of a system model is described using class diagram. Here class name is object detection and functions of object detection are detectObject, label bounding boxes, list bounding boxes, display image, load libraries, detect from image, detect from video. Attributes of object detection class are size, width, aspect ratio of image cell.

USE CASE DIAGRAM

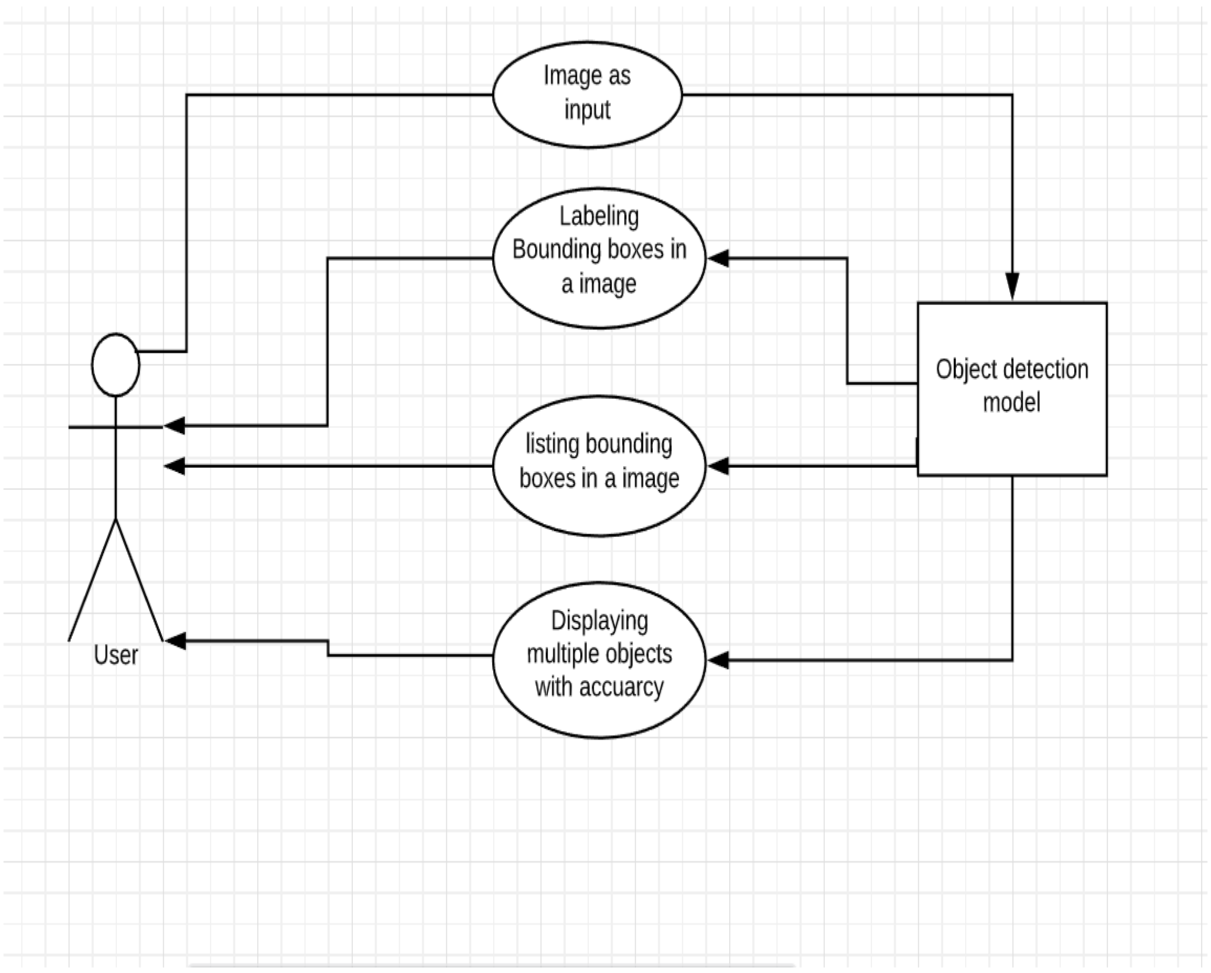
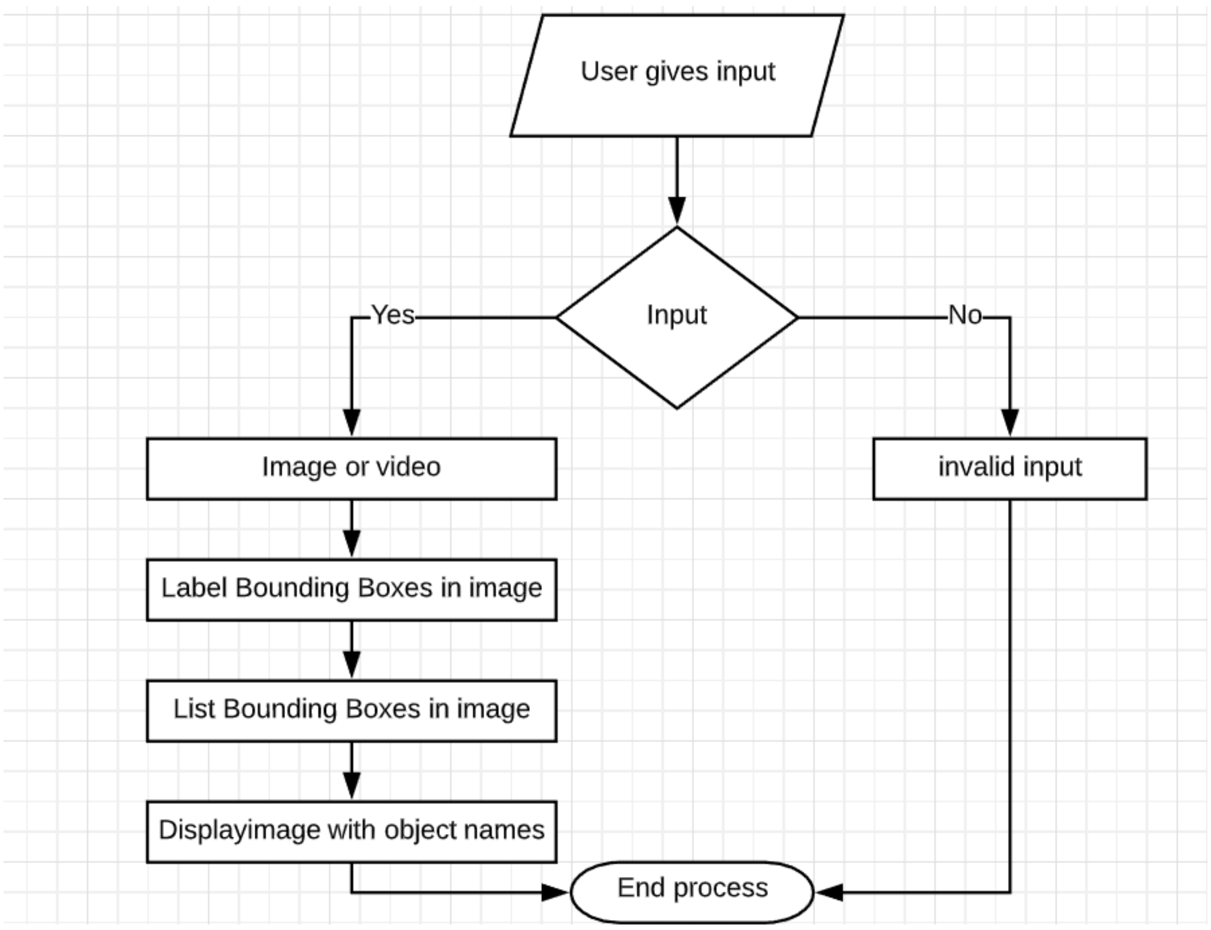


Image as input
Labeling Bounding boxes in a image
Object detection model
listing bounding boxes in a image
Displaying multiple objects with accuarcy
User

In above use case diagram, firstly user enters image as input. Next our object detection model scans the image and performs certain actions. It divides an image into n*n cells and labels the bounding boxes in an image. Next it lists all the label bounding boxes and finally displays output image by describing all the object names with accuracy.

DATA FLOW DIAGRAM


User gives input
Input
Yes
No
Image or video
invalid input
Label Bounding Boxes in image
List Bounding Boxes in image
Displayimage with object names
End process

The data flow diagram explains about flow of data. The way of input and out flow of data of a model is described by data flow diagram. It also gives information about inputs and outputs of each entity. Data flow diagrams not contains any loops and decision rules.

System Requirements:

Software requirements

- Pycharm ide
- Python (3.7.3)
- Anaconda prompt

Hardware Requirements:

- Intel core: 7th generation
- Ram- 8GB
- Rom- 512 GB

Make sure all the libraries are installed mainly cv2 to run the program.

Conclusion:

The given project worked well with no barriers in between. The project made us to learn many new things. Various libraries and their functions were clearly used by us now. The efficiency of the algorithm was high enough and the outputs were great. Learning of python by us has increased because of this project. The outcomes were great with no hurdles and all have worked very well. Computer vision course has developed lot of skills in us and generated new ideas and increased our scope of knowledge in the field of working with images and it’s processing. Deep learning networks usage have been to the core and yolo was highly used. I would like to thank Dr. Shan Du for her excellent teaching for us in the course throughout. She was very helpful in the teaching and clarified doughts whenever needed. We had a good course in computer vision.