SIP Transcript:

Thankyou apoorva..

Now let’s talk about helmet detection,

We have trained a YOLOv4 model to detect the presence of helmet for traffic monitoring.

This YOLO target detection algorithm is widely used in industry due to its high speed and high accuracy. As the backbone of YOLOv4, CSPDarknet-53 is responsible for extracting deep features of the input image.

Basically, the YOLO models are trained to look at an image and search for a subset of object classes. When found, these object classes are enclosed in a bounding box and their class is identified.

So in our case, when a video sequence is given as input, yolo is able to classify it as either with helmet class or without helmet class.

As we see in the image, a bounding box is drawn around the faces of riders and it is telling whether the person is with helmet or without helmet.

Our model is able to detect 3 types of helmets: like a full bike helmet, half helmet & cycle helmet

Coming to License plate detection,

This flowchart explains the working of license plate detection. So basically, the input image is a vehicle snippet which has violated some traffic rule, first we convert that into a gray scale image, and then image thresholding is done, and using contours we crop the number plate alone from the whole image, and then using tesseract module in python we segment and extract the characters from the license plate and then finally display our detected number plate charcters.

For ex, over here we’ve considered this car image, after applying few image processing techniques, and using contour detection, we r able to crop the number pate out as shown and then characters are also extracted and printed.

Coming to the project challenges and updates,

Whenever we were trying to detect just vehicles, objects that weren’t vehicles also were being detected. This problem was sorted by changing the masking method.

And also some of the number plates were unreadable, so for this pictures of vehicles were sorted instead.

Multiple vehicles speeds were unable to be detected.

This was sorted by saving speed timers in an array format instead of an single integer format

And when any 2 vehicles come close to each other, both of them are considered as a single object, so for that we have used eroded masked image to better differentiate close objects and also moved detection lines further back.

Coming to conclusion and future scope,

Most traffic accidents are caused by driver inattention to safety rules, distraction due to in-vehicle activities and fatigue.

The goal of the project is to successfully implement detection systems for keeping check of specific road safety rules being followed and make it easy for the traffic police department to monitor the traffic and take action against the violated vehicle owner in a fast and efficient way.

Some of the main causes of road accidents such as vehicles exceeding speed limit, helmet rule and jumping signal were planned to taken care off.

Further, a way for extracting the number plate details of the vehicles violating the rules has also been added. The designed algorithms were able effectively detect the type of violations and has proved to be efficient.

Furthermore, we are planning to extend our project by detecting the other traffic violations like wrong lane, wrong parking, etc.

SIP – helmet detection code:

Firstly, we r importing the required packages and modules

We created a class called video capture,

Where we r first importing yolo into variable called net using the files custom-yolo.weights file, custom-yolo.cfg file. so using this yolo we r going to detect helmets.

And we r loading another file called darknet.labels which contains 2 labels-one is with helmet and the other one without helmet, and then we store each of the label in classes list.

From net, we get out layers and r stored in output\_layers,

And then we r loading our input video.

Now in the process of detecting objects, we first use .blobFromImage fn to resize and crop image from center,

So using blob and net, we identify objects i.e whether there’s helmet or not on the person’s head.

And to show the information on screen, we first draw the bounding box around the head, so whenever the person’s head is detected, we find the centre point and then get the rectangle coordinates to draw the box.

And then we also label it as ‘with helmet’ or ‘without helmet’ in the video and also print it in terminal.

And finally, we return the output video

SIP – license plate detection code:

We first import packages and modules,

And then we r storing the car details like its name and number plate color, basically the number plate color can be either white or yellow. And there’s a case where the number plate bg is white as well as the car color is white, for that we have white\_bg.

We define a fn called img\_process which first converts the image to gray scale and applies threshold

And when the number plate color is white, we define ranges of white color in hsv form in lower\_white and upper\_white, similarly for yellow number plate. And then uses it to get our final processed image and stored as res\_img.

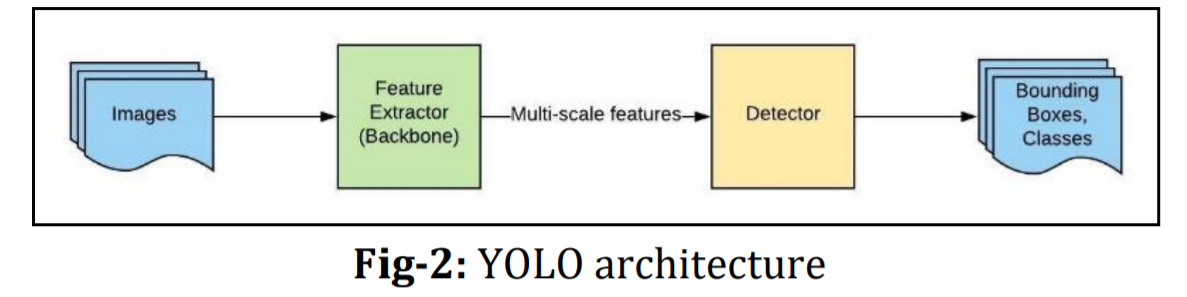
We define a fn called contour, basically this fn’s job is to crop out the number plate out of the processed image, if the color of the car and number plate is white we’re going to play special attention for that. If not we use, drawcontours fn to get the cropped image, and it is returned as crop,

we define show\_plt fn this uses pytesseract fn to convert the image to string, so that we can get the characters from the cropped number plate.

And this is our main fn, so first we Read the car images with visible number plates, Process the image by filtering using different threshold for white and yellow plates using img\_process fn., and using contour fn we crop the number plate, and finally using show\_plt fn, we extract number plate characters and print them.

first we convert that into a gray scale image, and then image thresholding is done, and using contours we crop the number plate alone from the whole image, and then using tesseract module in python we segment and extract the characters from the license plate and then finally display our detected number plate charcters.

Content:



5.2 YOLOv3 (You Only Look Once version 3) Fig-2: YOLO architecture In this model, a single neural network is applied to the concerned image. The image is then divided into multiple regions and probabilities are calculated and depending on their values bounding boxes are mapped. Non max suppression is deployed so that each object is detected only once and the overall accuracy of the model is increased. The output is then provided in the format that it shows the bounding box along with the appropriate class label and the probability calculated for that label. The version of YOLO used in the proposed system is version 3. A Darknet variant of 53 layers is used in this version in such a way that the architecture of this model consists of 106 fully functional convolutional layers

RV Transcript:

Coming to the Results,

When we loaded this cardboard image as input, our CNN model was able to classify it as a cardboard object with 77% probability, and we also obtained the probability of it being other objects too like glass, paper, plastic, trash, metal and cardboard.

So, when we plotted all these probabilities on a graph, we got the max value for cardboard, so our model classified it as a cardboard object, and then the 2nd highest percent is 17% for plastic, the reason for the model predicting as plastic can be because of the tape pasted on it.

Similarly, for the bellow bottle image, it’s classified as glass object with almost 80% prob, and it is also detected as plastic with 10% prob, so may be the reason for it detecting as plastic might be because of the sticker present on the bottle.

Coming to conclusion,

The classification of trash into various waste categories is possible through machine learning and computer vision algorithms. One of the biggest pain point is the wide varieties of possible data (i.e. any object can be classified into one of the waste or recycling categories). Therefore, in order to create a more accurate system, we have considered a large and continuously growing data source.

So, finally our CNN model was able to predict the probability of the object being either a glass or paper or plastic or trash or metal or cardboard with almost 80% accuracy and is classify the object as one of them which has maximum probability.

Coming to future scope,

Furthermore, we would like to extend this project to identify and classify multiple objects from a single image or video. This could help recycling facilities more by processing a stream of recycling rather than just single objects. Finally, we want to expand our dataset by adding more photos for a even more better performance, and extend these 6 classes to more classes.

RV Code:

So as we trained the prediction model, now let’s test it with images.

So first we r loading image and converted the image into integer array and then were displaying the image using .squeeze() fn so that we can remove single-dimensional entries from the shape of an array.

Now using .predict fn we r predicting the probabilities and stored it in array p.

Now we’re printing out the max probability in p and the corresponding label for it, so over here our max prob is 0.524 and the corresponding label is cardboard.

Now let’s look at the individual probabilities for 6 labels,

So these are the probability percentages and we’ve also plotted our probabilities vs labels in a graph for a better understanding.

Coming to accuracy,

As apoorva has already trained our model and stored it as history, using that we get the training accuracy and validation accuracy values and we’re plotting the graph, similarly we r also plotting for training loss and validation loss.