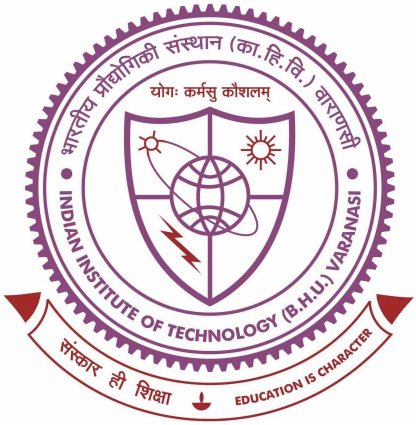
DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY

(BANARAS HINDU UNIVERSITY)

VARANASI - 221005



EXPLORATORY PROJECT REPORT

on

SELF DRIVING CAR

(Using deep learning)

by

Tanvi – (20085102)

Muskan – (20085057)

Lavi Goyal – (20085050)

Under the supervision of

Prof. Rakesh Kumar Mishra

Acknowledgement

We would like to express our gratitude to Prof Rakesh Kumar Mishra for their able guidance and support in completing our exploratory project on Self driving Car. We would also like to thank our parents and friends for their continuous support during the project.

Contents :

AIM:

Detection of Lane, objects, and traffic signals during autonomous driving.

ABSTRACT:

- A self-driving car, also known as an autonomous car or driverless car, is a vehicle that can travel between destinations without the need of any human effort.
- They can process streams of data from different sensors such as cameras, LiDAR, RADAR, GPS, or inertia sensors. This data is then modeled using deep learning algorithms, which then make decisions relevant to the environment the car is in.
- This report consists of three sections:
    1. Road lane detection using Computer Vision.
    2. Object detection using YOLOv3 algorithm.
    3. Traffic signs detection using Convolutional Neural Networks (CNN).

Libraries / Tools Used:

- Python
- OpenCV
- Tensorflow
- Matplotlib
- Keras
- Numpy
- Pandas
- Scikit-learn

Self Driving Car and Their Working :

How self-driving cars make decisions?

Driverless cars can identify objects, interpret situations, and make decisions based on object detection and object classification algorithms. They do this by detecting objects, classifying them, and analyzing what they are.

How does a self-driving car see?

The three significant sensors used by self-driving cars work together as the human eyes and brain. These sensors are cameras, radar, and lidar. Together, they give the vehicle a clear view of its environment. They help the car identify the location, speed, and 3D shapes of objects close to it. Additionally, self-driving cars are now being built with inertial measurement units that monitor and control acceleration and location.

- **Reliable cameras**

Self-driving cars have several cameras at every angle for a perfect view of their surroundings. While some cameras have a broader field of view of about 120 degrees, others have a narrower view for long-distance vision. Fish-eye cameras provide extensive visuals for parking purposes.

- **Radar detectors**

Radar detectors augment the efforts of camera sensors at night or whenever visibility is poor. They send pulses of radio waves to locate an object and send back signals about the speed and location of that object.

- **Laser focus**

Lidar sensors calculate distance through pulsed lasers by empowering driverless cars with 3D visuals of their surroundings, adding richer information about shape and depth.

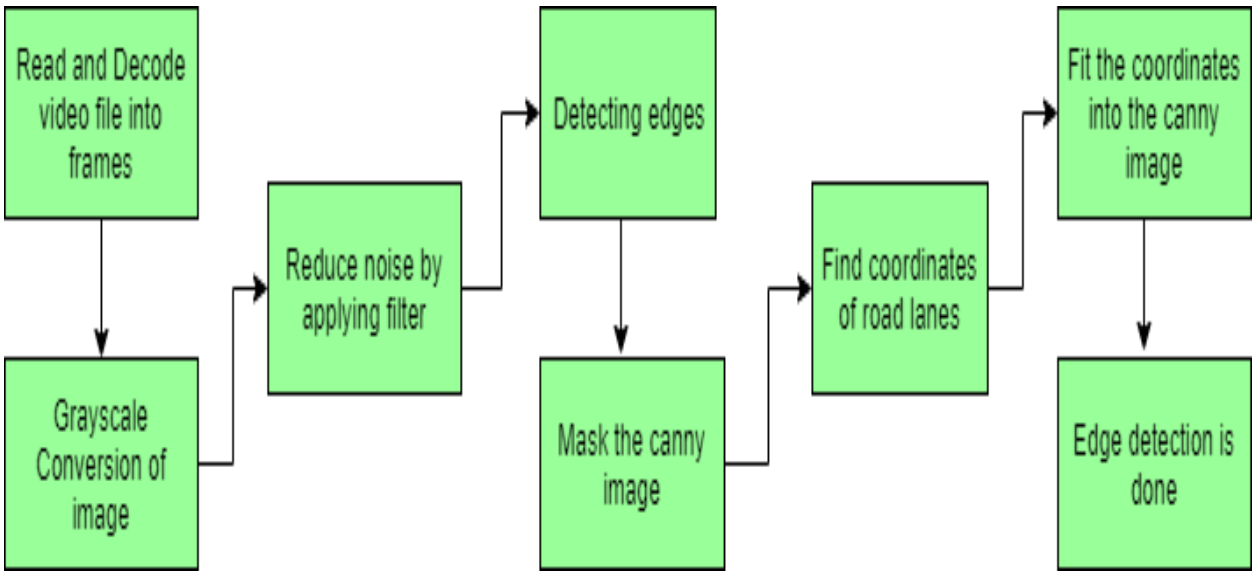
- **LiDAR**

LiDAR is one of the most important technologies used to develop self-driving vehicles. It is a device that sends out pulses of light that bounce off an object and returns to the LiDAR sensor, which determines its distance. The LiDAR produces a 3D Point Cloud, a digital representation of how the car sees the physical world.

TASK : LANE DETECTION

One of the many steps involved during the training of an autonomous driving car is lane detection, which is the preliminary step.

Lane detection involves the following steps which can be understood through this flow diagram :



Now we will take a each of these steps one by one to understand their process and procedure :

- **Capturing and decoding video file:** We will capture the video using VideoCapture object and after the capturing has been initialized every video frame is decoded (i.e., converting into a sequence of images).

(Please turn to next page to see video frame depicting above mentioned step)

Video frame (at some moment):

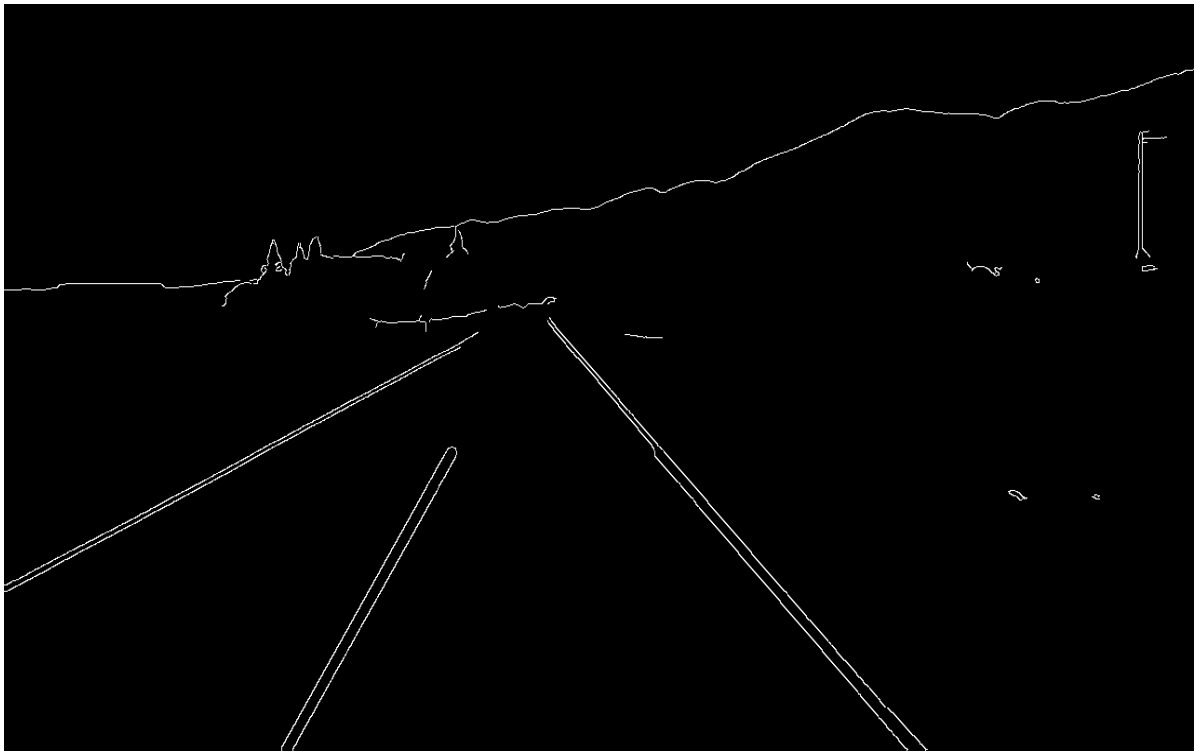
- **Grayscale conversion of image:**

The video frames are in RGB format, RGB is converted to grayscale because processing a single channel image is faster than processing a three-channel colored image.

- **Reduce noise:**

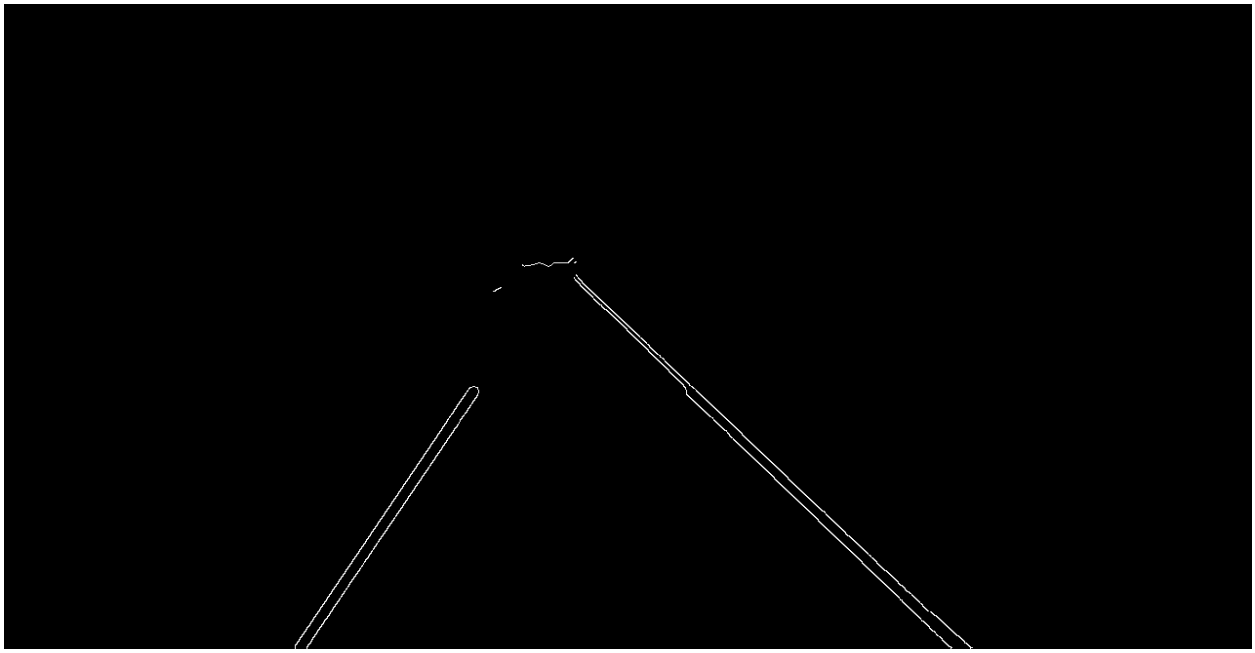
Noise can create false edges, therefore before going further, it's imperative to perform image smoothening. Gaussian filter is used to perform this process.

- **Canny Edge Detector:** It computes gradient in all directions of our blurred image and traces the edges with large changes in intensity.

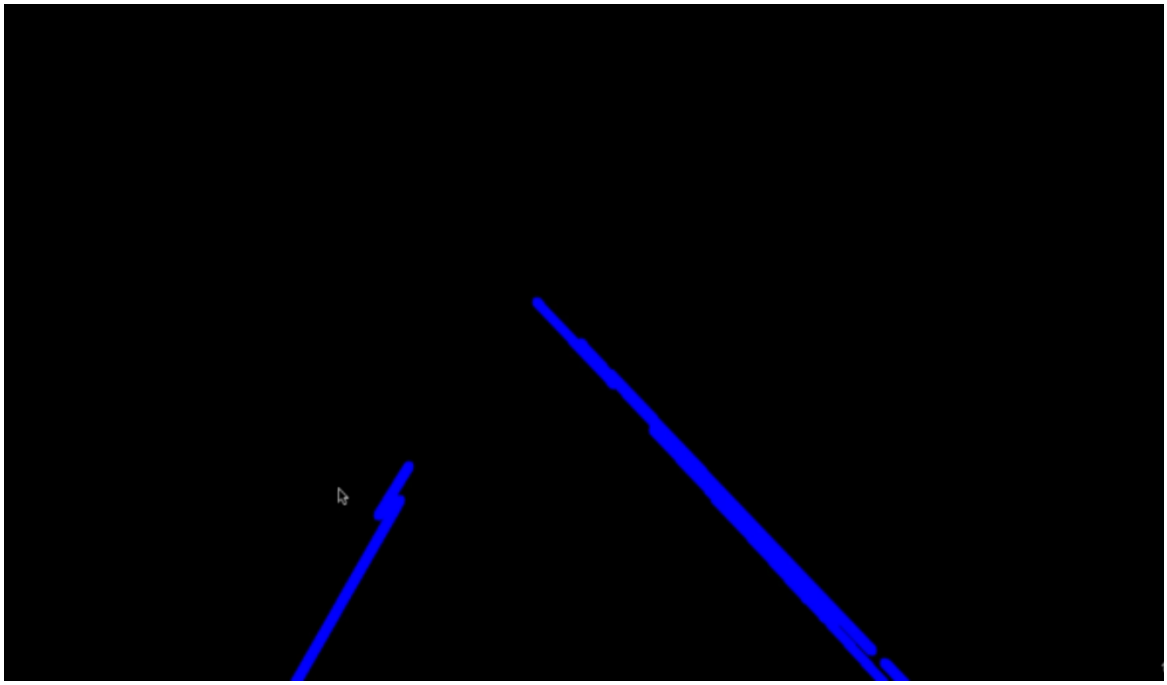


- **Region of Interest:**

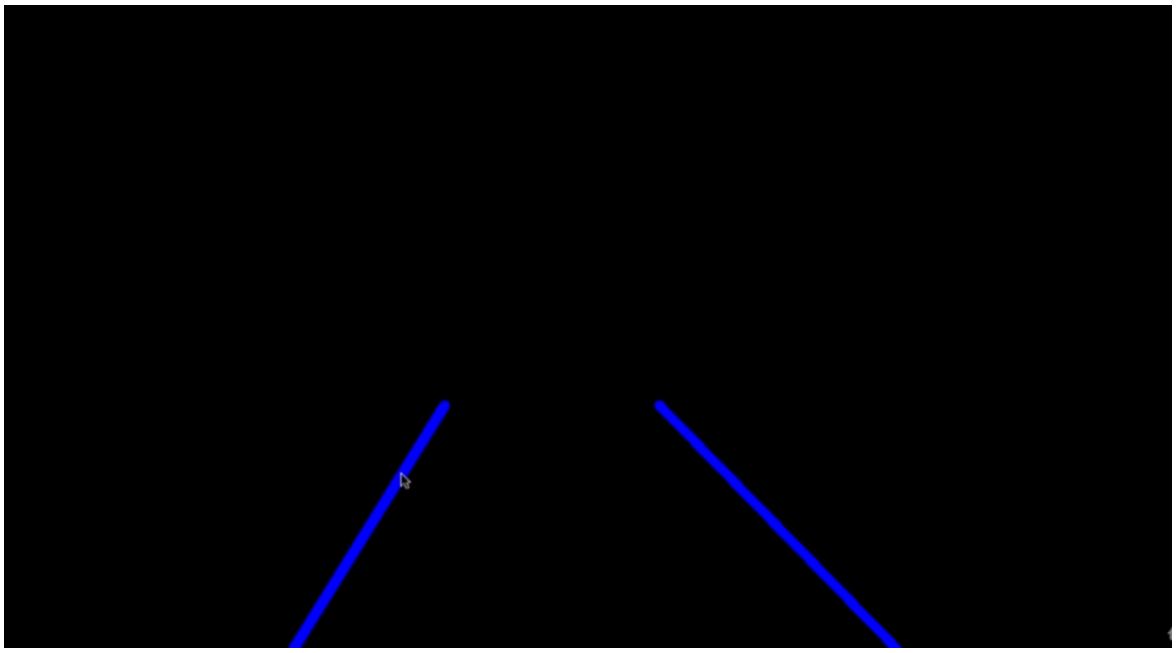
This step is to take into account only the region covered by the road lane. A mask is created here, which is of the same dimension as our road image. Furthermore, bitwise AND operation is performed between each pixel of our canny image and this mask. It ultimately masks the canny image and shows the region of interest traced by the polygonal contour of the mask.



- **Hough Line Transform:** The Hough Line Transform is a transform used to detect straight lines. The Probabilistic Hough Line Transform is used here, which gives output as the extremes of the detected lines.



- **Averaged - Lines**: Now, instead of having multiple lines obtained from the Hough line transform technique, average of all the slopes and y- intercepts of these lines is calculated to get a single line for both our left and right lane.



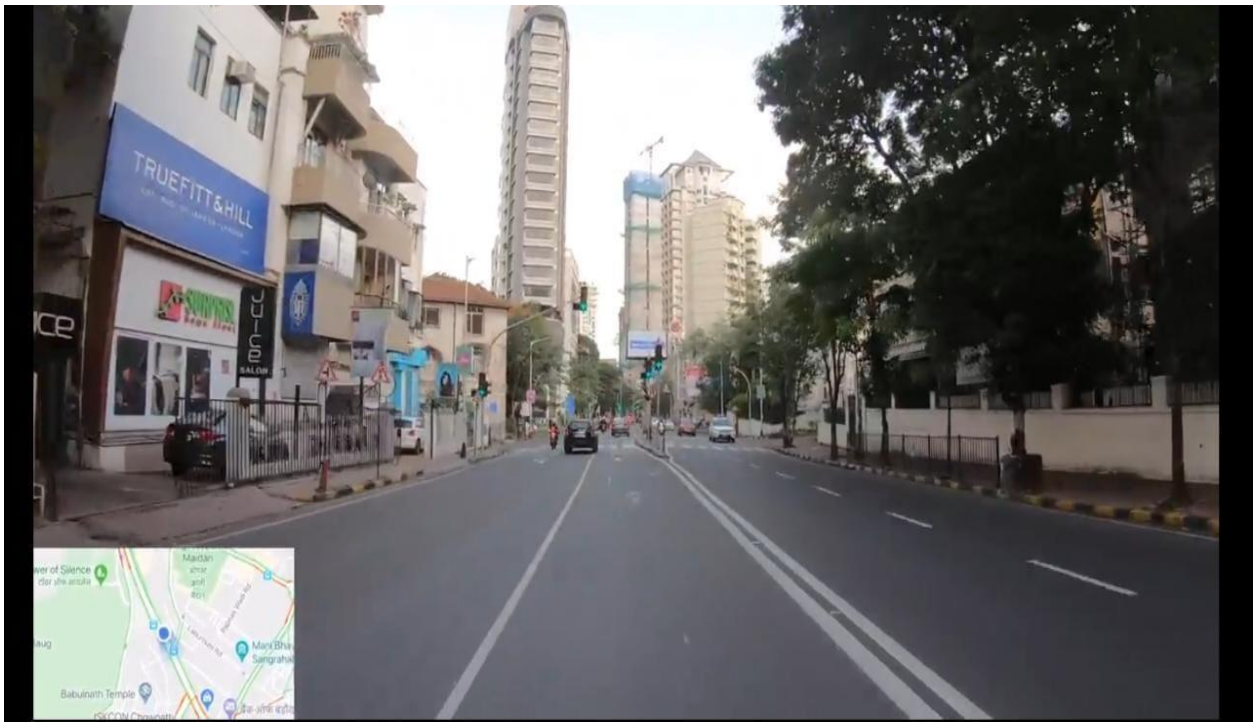
Final image after blending the averaged-lines with the initial image:

Evaluation:

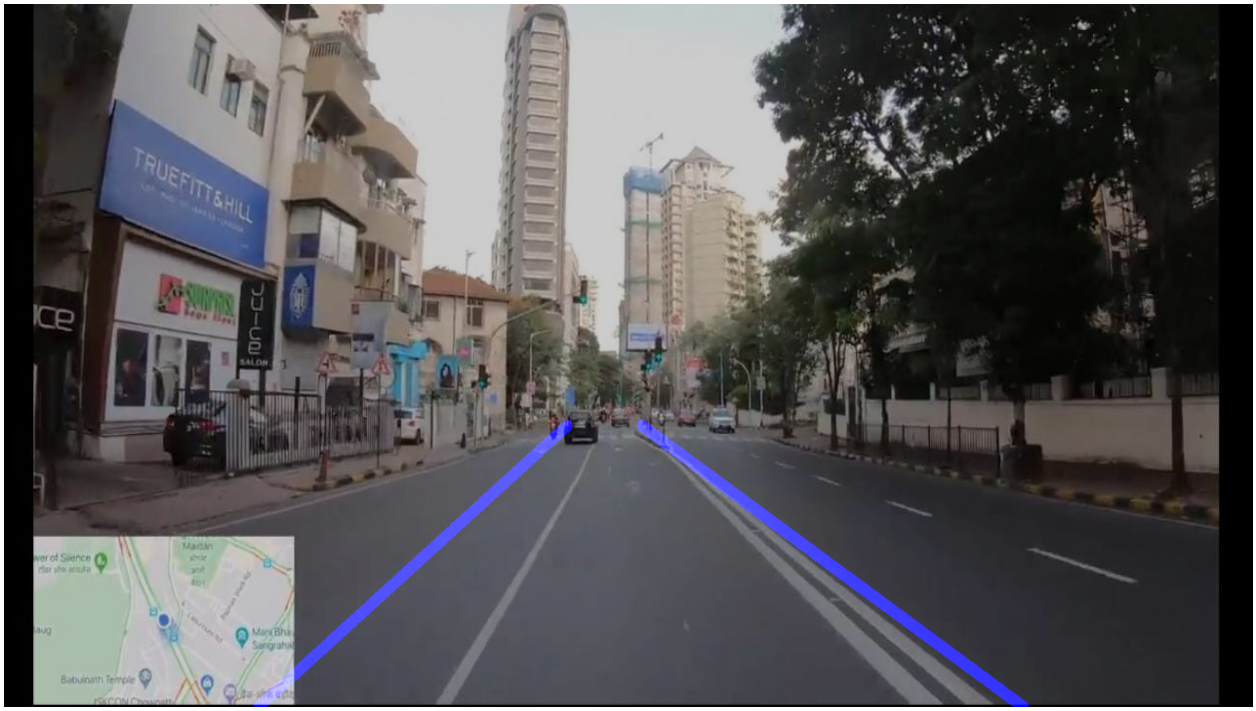
INPUT IMAGE 1:

OUTPUT IMAGE 1:

INPUT IMAGE 2:



OUTPUT IMAGE 2:



INPUT IMAGE 3:

OUTPUT IMAGE 3:

TASK : OBJECT DETECTION

Object detection is also one of the critical components to support autonomous driving. Autonomous vehicles rely on the perception of their surroundings to ensure safe and robust driving performance. This perception system uses object detection algorithms to accurately determine objects such as pedestrians, vehicles, traffic signs, and barriers in the vehicle's vicinity

Solution: We have used YOLOv3 along with open cv to detect objects

What is YOLO ?

- YOLO (You Only Look Once) is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.
- YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.
- This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.
- The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

Benefits of YOLO:

It has high Speed, High accuracy and Learning capabilities

Architecture:

This architecture takes an image as input and resizes it to 448*448 by keeping the aspect ratio the same and performing padding. This image is then passed in the CNN network. This model has 24 convolution layers and 4 max-pooling layers followed by 2 fully connected layers. To reduce the number of layers (Channels), we use 1*1 convolution followed by 3*3 convolution.

This is done by generating (1, 1470) from the final fully connected layer and reshaping it to size (7, 7, 30).

This architecture uses Leaky ReLU as its activation function in the whole architecture except for the layer where it uses a linear activation function.

WORKING:

YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

RESIDUAL BLOCKS :

•First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.

•In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

BOUNDING BOX REGRESSION :

•A bounding box is an outline that highlights an object in an image.

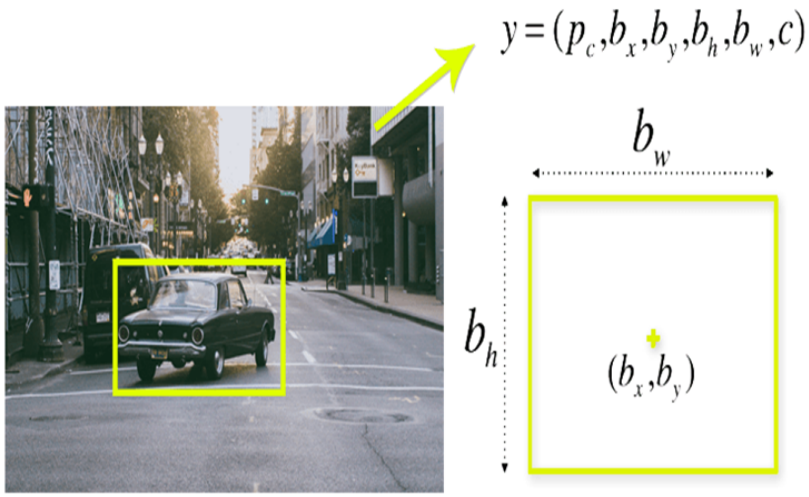
•Every bounding box in the image consists of the following attributes:

•Width (bw) and Height (bh).

•Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.

•Bounding box center (bx,by).

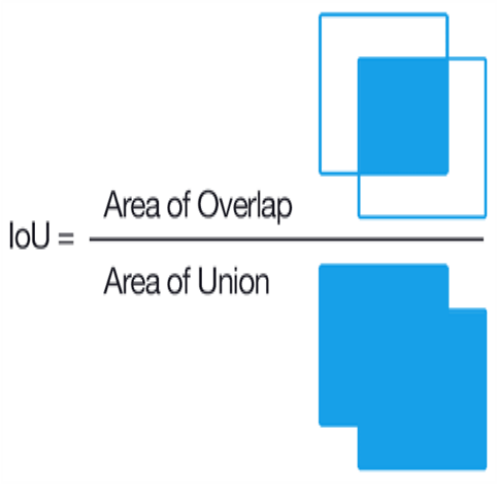
•The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.

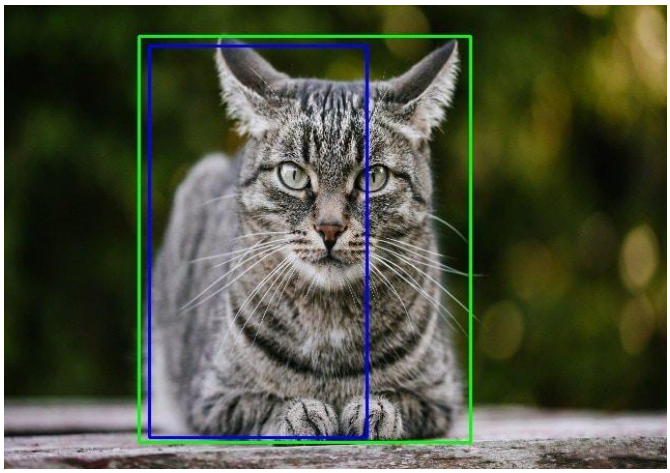


•YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

INTERSECTION OVER UNION (IOU) :

1. Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.
2. Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.
3. The following image provides a simple example of how IOU works. In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.





How does the YOLO Framework Function?

It passes the labelled data to the model in order to train it. Suppose we have divided the image into a grid of size 3 X 3 and there are a total of 3 classes which we want the objects to be classified into. Let's say the classes are Pedestrian, Car, and Motorcycle respectively. So, for each grid cell, the label y will be an eight-dimensional vector:

Here,

- p_c defines whether an object is present in the grid or not (it is the probability)
- b_x, b_y, b_h, b_w specify the bounding box if there is an object
- c_1, c_2, c_3 represent the classes. So, if the object is a car, c_2 will be 1 and c_1 & c_3 will be 0, and so on

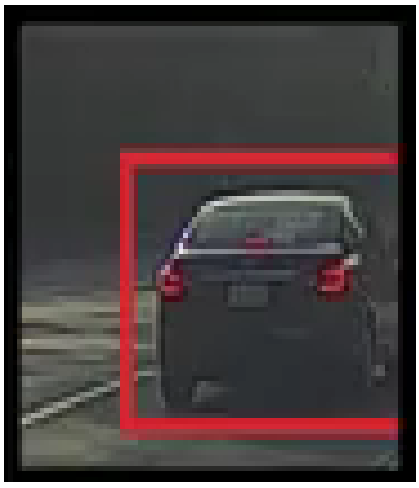
$$y = \begin{bmatrix} pc \\ bx \\ by \\ bh \\ bw \\ c1 \\ c2 \\ c3 \end{bmatrix}$$

- Let's say we select the first grid from the above example. Since there is no object in this grid, pc will be zero and the y label for this grid will be:

Here, '?' means that it doesn't matter what b_x, b_y, b_h, b_w, c_1, c_2, and c_3 contain as there is no object in the grid.

y =	0
	?
	?
	?
	?
	?
	?
	?

- Let's take another grid in which we have a car ($c_2 = 1$).
  Before we write the y label for this grid, it's important to first understand how YOLO decides whether there actually is an object in the grid. In the above image, there are two objects (two cars), so YOLO will take the mid-point of these two objects and these objects will be assigned to the grid which contains the mid-point of these objects. The y label for the center left grid with the car is:



y =	1
	bx
	by
	bh
	bw
	0
	1
	0

Calculating b_x, b_y, b_h, b_w

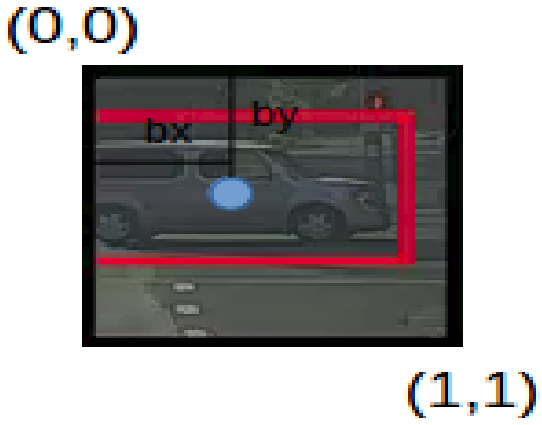
- b_x, b_y, b_h, and b_w will be calculated relative to this grid only. The y label for this grid will be:

y =	1
	bx
	by
	bh
	bw
	0
	1
	0

- $p_c = 1$ since there is an object in this grid and since it is a car, $c_2 = 1$

- b_x, b_y are the x and y coordinates of the midpoint of the object with respect to this grid. In this case, it will be (around) $b_x = 0.4$ and $b_y = 0.3$:

- b_h is the ratio of the height of the bounding box (red box in the above example) to the height of the corresponding grid cell, which in our case is around 0.9. So, $b_h = 0.9$. b_w is the ratio of the width of the bounding box to the width of the grid cell. So, $b_w = 0.5$ (approximately).



y =	1
	0.4
	0.3
	0.9
	0.5
	0
	1
	0

NON-MAX SUPPRESSION :

- There is one more technique that can improve the output of YOLO significantly – **Non-Max Suppression.**

- One of the most common problems with object detection algorithms is that rather than detecting an object just once, they might detect it multiple times. Consider this image:

- Here, the cars are identified more than once. The Non-Max Suppression technique cleans this up so that we get only a single detection per object.

Evaluation:

Input image 1:

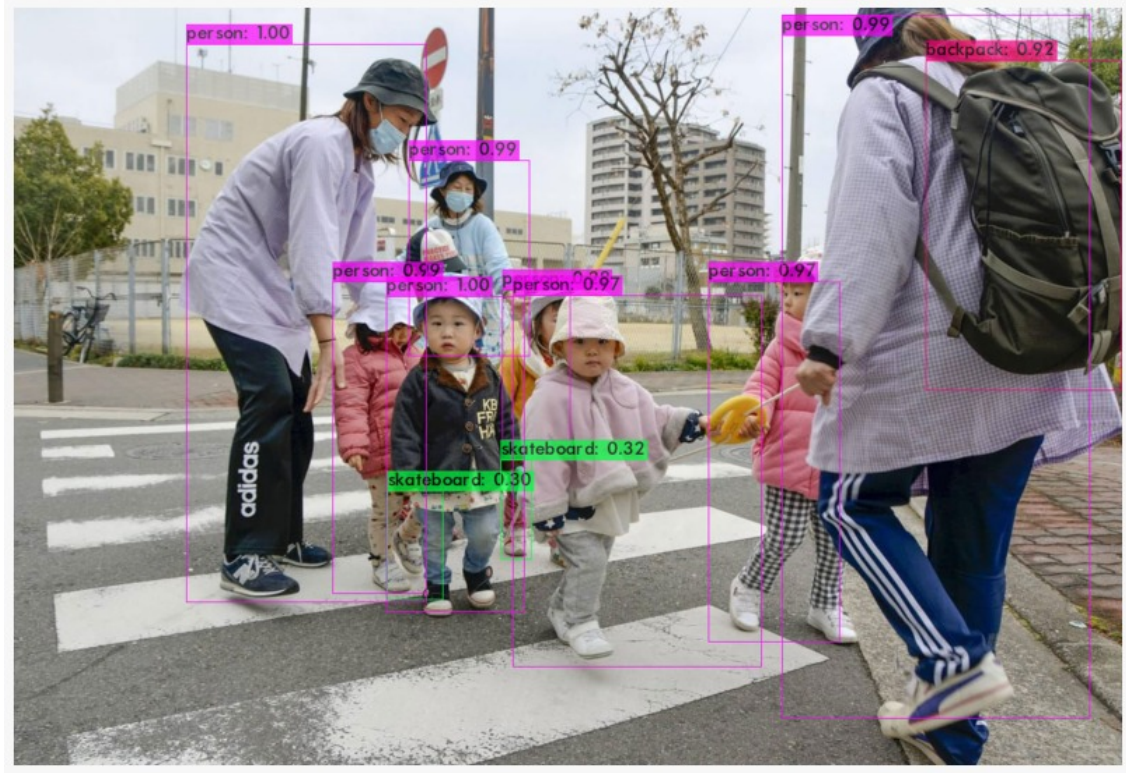
Output image 1:

Input image 2:

Output image 2:

Input image 3:

Output image 3:



Input image 4:

Output image 4:

TASK: ROAD SIGNS DETECTION AND RECOGNITION

Traffic-sign recognition is a challenging subject and also a valuable subject in traffic engineering research. Traffic signs contain necessary messages about vehicle safety and they show the latest traffic conditions, define road rights, forbid and allow some behaviors and driving routes, etc.

Solution:

We have used CNN based on a transfer of learning method for detection of traffic signs. Deep CNN is trained with a big data set, and then effective regional convolutional neural network (RCNN) detection is obtained through a spot of standard traffic training examples.

What is CNN?

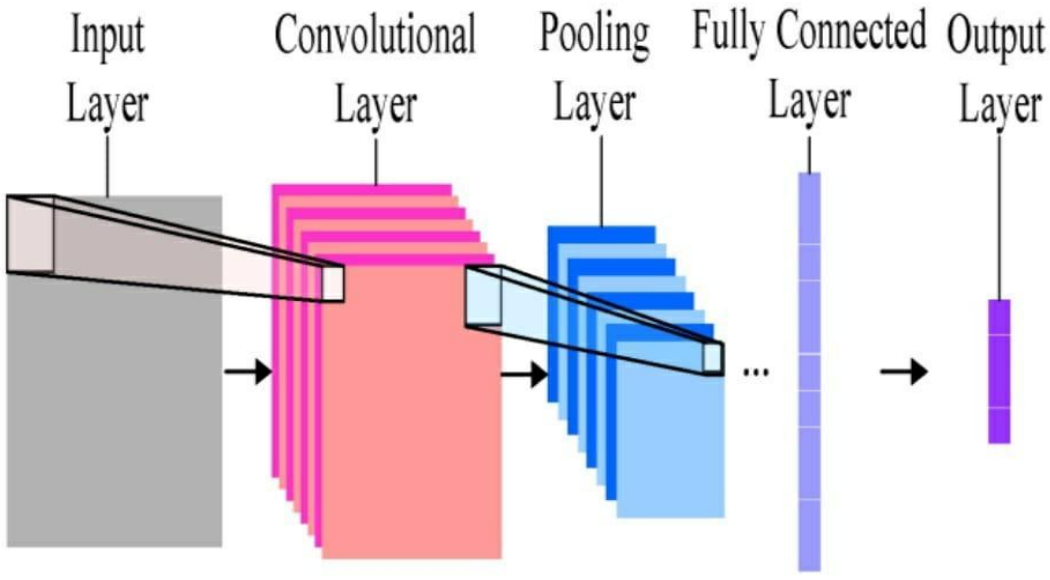
A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

CNN is very similar to the brain as it also has neurons which in turn have weights and biases. Each neuron receives an input on which it performs some operation, and the output is passed as the input to the next neuron. It can have many layers, the first is the input layer, and the last is the output layer. Anything else in between is called a hidden layer.

Working of CNN:

CNN every image is represented in the form of pixel values and it compares images piece by piece. It is commonly used to examine visual pictures by handling information with a grid-like topology.

CNN has the following layers:



If any layers fail to perform their task, then the process will never be executed.

In the convolutional layer, we first line up the feature along with the picture and then multiply the pixel value with the corresponding value of the filter and then add them up and drive them with the absolute value of pixels.

ReLU layer represents a rectified layer unit crafted by this layer to expel all the negative qualities from the filter picture and afterward change it to zero.

The node is activated if the image is above a certain quality, and it is linearly dependent on the variable considering the input is above a threshold value.

The image received from Relu is shrunk in the pooling layer. The actual classification is done in a fully connected layer.

We take the shrieked image and up that in a single list. And then, we compare that image with our previously-stored list and judge the image. The last is the output layer which gives the output of the classified image.

Traffic sign detection and recognition working:

1.) PHASE 1: DETECTION

In this phase, the image obtained from the camera in the car is preprocessed before detection starts. General preprocessing steps involve converting the obtained RGB image into an HSV image.

The HSV (Hue Saturation Value) color space is preferred over the RGB (Red Green Blue) color space for detection. HSV is more similar to what the human eye sees when compared to an RGB image.

Once the HSV image is obtained, the following steps would be to detect objects based on their color, determine their shape, and validate the object to be a traffic sign.

2.) PHASE 2: RECOGNITION

Once the sign is detected, the sign must also be classified. With the help of TensorFlow and CNN can be implemented. Training and testing is the most crucial part of this phase.

We used the German Traffic Sign Benchmark data set for training and testing. In this, a feedforward network has six convolutional layers. This also has fully connected hidden layers with dropout layers to prevent overfitting.

The model uses the sequential stack provided by Keras, an open-source high-level neural network library capable of running on top of TensorFlow. All the layers of the proposed CNN have Rectified Linear Unit (ReLu) activation.

The output of the 6th convolutional layer is fed to a fully connected layer, which uses a flatten function to flatten the output at that point. The flattened output is given to the final layer, which uses SoftMax activation.

After every two convolutional layers, a max-pooling layer is added to improve processing speed.

We do not use just one CNN but an ensemble of CNNs. We have 3 CNN's whose aggregate determines the final result. This provides a much more accurate result than just a single CNN. The loss, optimizer, and metrics of the model were observed.

The loss of the model is set to categorical cross-entropy, and adam is used as an optimizer. The metric used is accuracy. Epochs with a backward pass are used to increase the accuracy of the prediction.

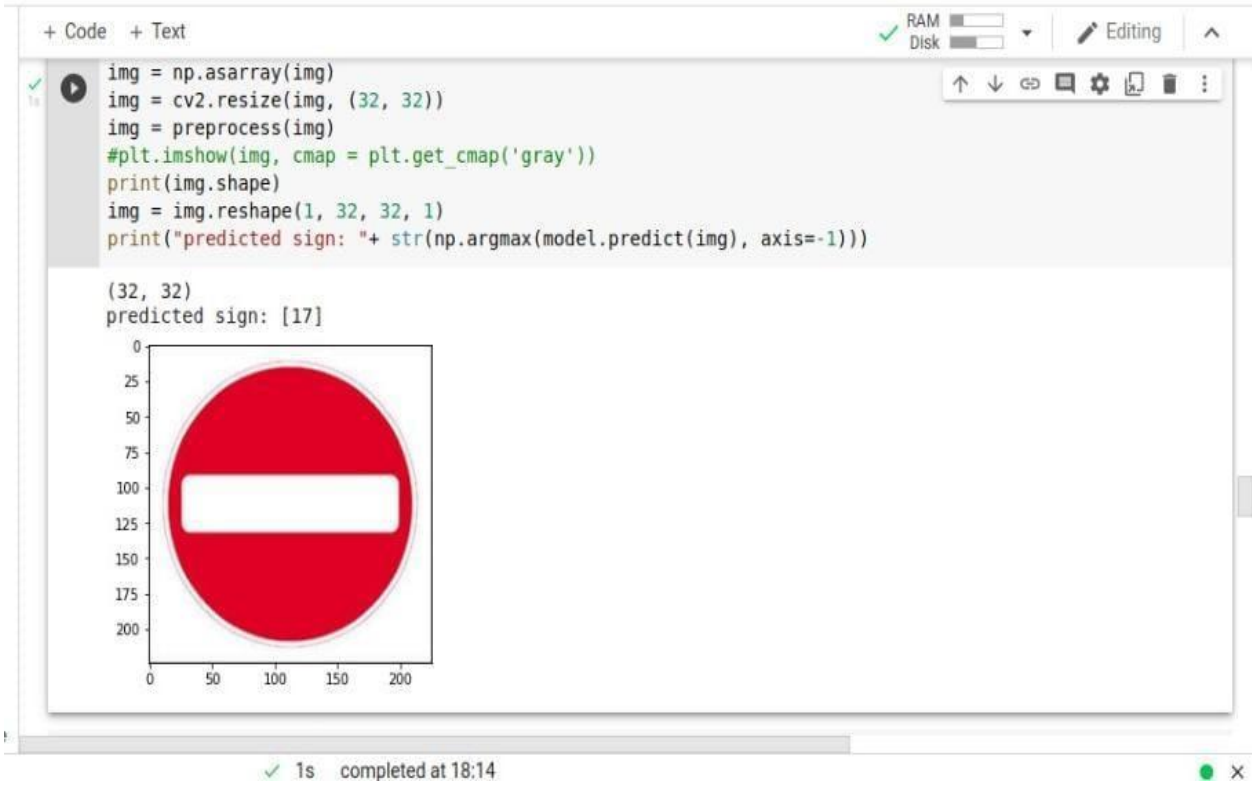
We first convert the image into the dimension of shape (1, 30, 30, 3) because to predict the traffic sign, the dimension must be the same with the dimensions we have used when building the model. Then it predicts the class, and the model.predict classes(image) returns us a number between (0-42) representing the class it belongs to. We use the dictionary to get the information about the class.

Evaluation:

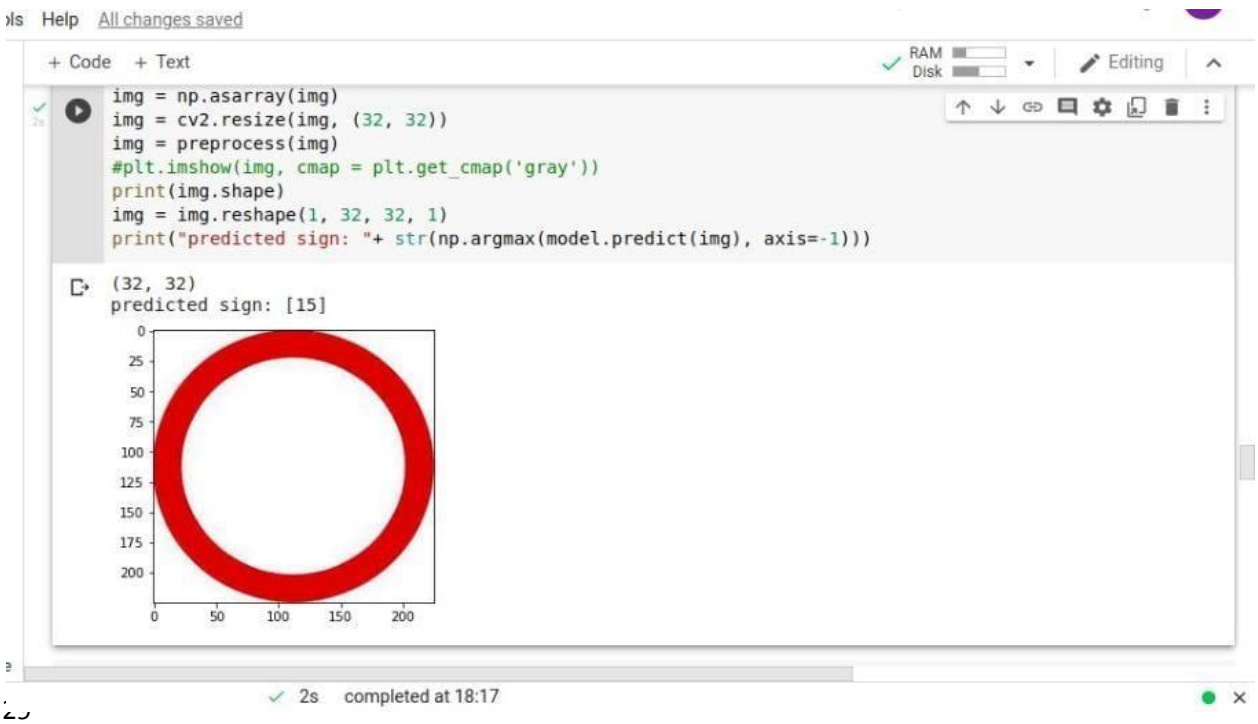
ClassId and SignName of Traffic signs used:

```
ClassId,SignName
0,Speed limit (20km/h)
1,Speed limit (30km/h)
2,Speed limit (50km/h)
3,Speed limit (60km/h)
4,Speed limit (70km/h)
5,Speed limit (80km/h)
6,End of speed limit (80km/h)
7,Speed limit (100km/h)
8,Speed limit (120km/h)
9,No passing
10,No passing for vechiles over 3.5 metric tons
11,Right-of-way at the next intersection
12,Priority road
13,Yield
14,Stop
15,No vechiles
16,Vechiles over 3.5 metric tons prohibited
17,No entry
18,General caution
19,Dangerous curve to the left
20,Dangerous curve to the right
21,Double curve
22,Bumpy road
23,Slippery road
24,Road narrows on the right
25,Road work
26,Traffic signals
27,Pedestrians
28,Children crossing
29,Bicycles crossing
30,Beware of ice/snow
31,Wild animals crossing
32,End of all speed and passing limits
33,Turn right ahead
34,Turn left ahead
35,Ahead only
36,Go straight or right
37,Go straight or left
38,Keep right
39,Keep left
40,Roundabout mandatory
41,End of no passing
42,End of no passing by vechiles over 3.5 metric tons
```

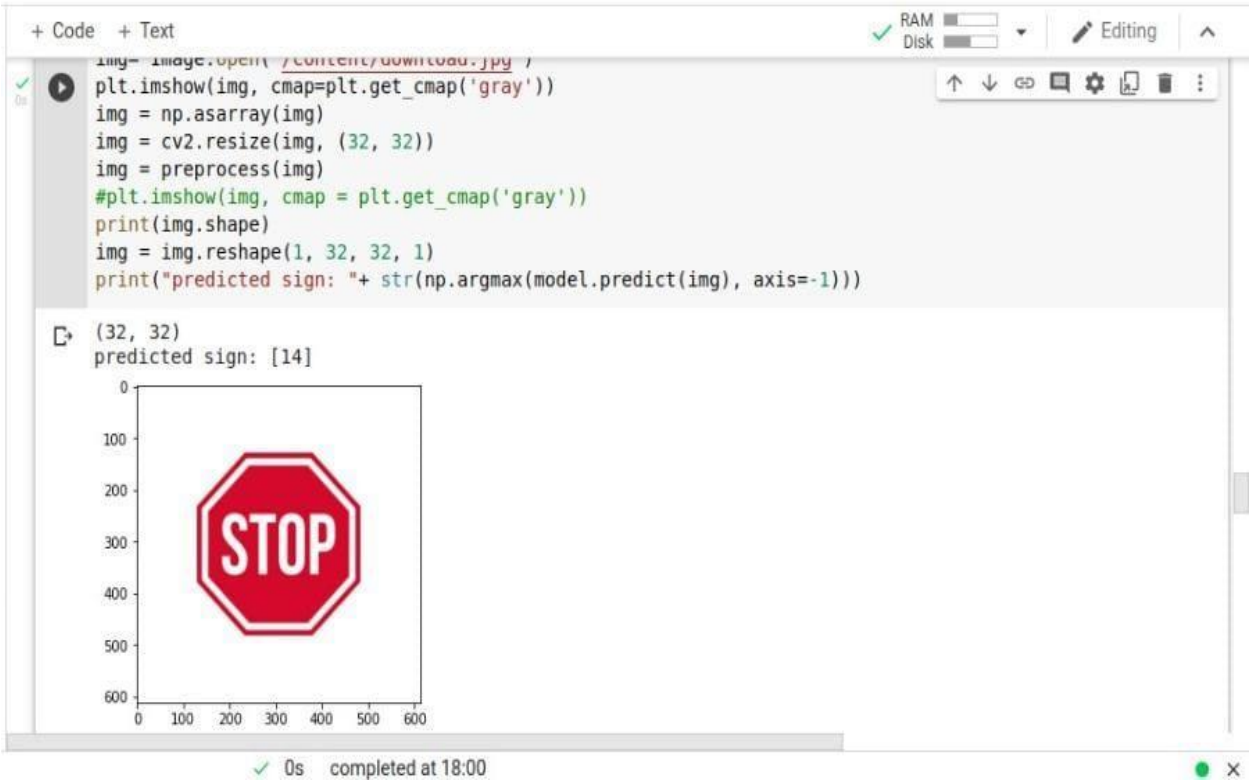
Sample image 1.



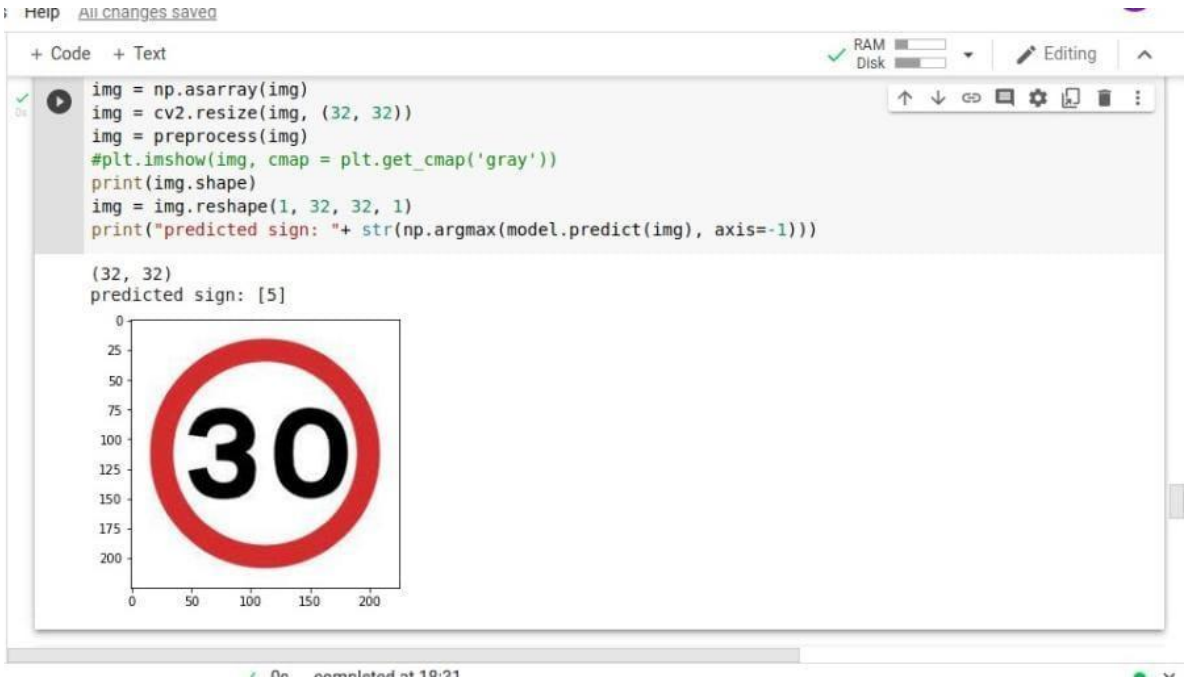
Sample image 2.



Sample image 3.



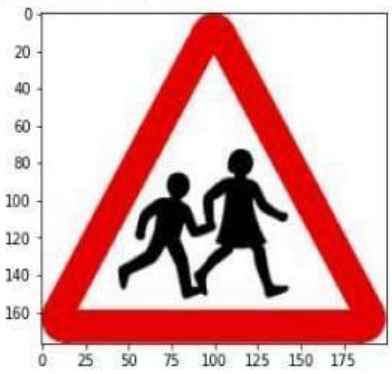
Sample image 4.



Sample image 5.

```
img = np.asarray(img)
img = cv2.resize(img, (32, 32))
img = preprocess(img)
#plt.imshow(img, cmap = plt.get_cmap('gray'))
print(img.shape)
img = img.reshape(1, 32, 32, 1)
print("predicted sign: "+ str(np.argmax(model.predict(img), axis=-1)))
```

```
(32, 32)
predicted sign: [28]
```



✓ 0s completed at 18:29

Results and Conclusion:

Road lanes, obstacles around the car and traffic signs detection is a requisite part of autonomous driving.

Lane detection which was done using computer vision identified the lanes properly.

Yolo v3 algorithm which was used for object detection has high speed, accuracy and learning capabilities. However, the algorithm struggles to detect small objects and low-resolution objects. Moreover, the yolov3 trained model may not be ideal when large datasets are hard to obtain.

Accuracy of 98.9% was achieved using the CNN trained model for traffic sign detection. One of the limitations of the model is that it cannot be trained on a different dimension of images. So it is mandatory to have same dimension in the dataset. Another limitation is that due to a large number of classes it was not able to predict a few traffic signs correctly.

Autonomous vehicle technology is not yet mature enough. Still, it needs rigorous exposure to a wide range of traffic, landscape, and natural conditions in which the autonomous vehicles can be trained to perform as expected in actual traffic conditions.

References:

1. Joseph Redmon, Ali Farhadi, “YOLOv3: An Incremental Improvement”, University of Washington.
2. Karlijn Alderliesten, “YOLOv3 — Real-time object detection”, May 28 2020
3. Research paper on “Object Detection and Classification using YOLOv3” published by https://www.ijert.org/
4. The Complete Self driving car course- Applied deep learning by Udemy https://www.udemy.com/course/applied-deep-learningtm-the-complete-self-driving-car-course/
5. Arka Prava Jana, Abhiraj Biswas, Mohana, “YOLO based Detection and Classification of Objects in video records” ,2018 IEEE International Conference
6. https://learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/
7. https://towardsdatascience.com/lane-detection-with-deep-learning-part-1-9e096f3320b7
8. https://towardsdatascience.com/lane-detection-with-deep-learning-part-2-3ba559b5c5af
9. Machine learning course on Coursera by Andrew Ng https://medium.com/swlh/behavioural-cloning-end-to-end-learning-for-self-driving-cars-50b959708e59
10. Md. Rezwanul Haque, Md. Milon Islam, Kazi Saeed Alam, Hasib Iqbal, Md. Ebrahim Shaik, “A Computer Vision based Lane Detection Approach", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.11, No.3, pp. 27-34, 2019.