JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

2020-2021

MINOR II

FINAL EVALUATION REPORT

AUTONOMOUS CAR USING DEEP LEARNING

GROUP MEMBERS -

AKASH SONI	17103029	B 1
SUYASH VERMA	17103052	B 10
VAIBHAV MISHRA	17103346	B 9

OUTLINE

PROBLEM STATEMENT

Recently the number of vehicles on the road has been increased enormously thanks to the technological achievement of the motor industry and very precisely the availability of motor vehicles at very low rates. With this remarkable growth of vehicles on the road there is an enormous increase in the road accidents as well and the ignorance of traffic signs is considered as the major cause. Nearly 1.25 million people dies of road accident yearly and about 20-50 million are injured or disabled.

DoT researchers estimates that fully autonomous vehicles could reduce the traffic fatalities by up to 94 percent by eliminating those accidents that are caused by Human errors. So millions of lives can be saved every year using Autonomous cars.

Cars with automated technology have sensors that never lose vigilance. "They are always looking for pedestrians. They are always looking for edge of the road. They are always watching the car in front. They don't become distracted or drunk, and I think that's really the main reason why most experts would say that there is definite possibility that automation can significantly reduce those Human error caused fatal crashes".

The biggest problem automated vehicles solve is that human beings are not well suited to travelling at high speeds. As speed increases, our time and distance perception degrades. We do not have sufficient range of lateral vision and no rear vision. We are limited in on ability to focus and process data and we tend to focus on one thing at a time (which leads to distraction from the task of driving). In short, we make mistakes behind the wheel and these mistakes are primary cause in more than 90% of accidents.

Abstract

For vehicles to be able to drive by themselves, they need to understands their surrounding world like human drivers, so that they can navigate their way in the streets, pause at traffic signs and traffic lights and avoid hitting obstacles such as others cars and pedestrians.
Autonomous Driving Car is one of the most disruptive innovations in AI. Fuelled by Deep Learning algorithms, they are continuously driving our society forward and creating new opportunities in the mobility sector. An autonomous car can go anywhere a traditional car can go and does everything that an experienced human driver does. But it’s very essential to train it properly. One of the many steps involved during the training of an autonomous driving car is lane detection.

Objectives

In our Minor Project II we are trying to solve above two problems of an Autonomous car

1. Traffic Sign Recognition and
2. Lane detection

In this Project we have implemented Traffic sign recognition system using Convolutional neural network and
Automatic Lane detection system using Computer Vision techniques via opencv

Literature Review

S.R No	Paper Details (IEEE Style)	Year	Objectives	Strength	Weakness	Result Obtained
1.	Pannu, Gurjashan Singh, Mohammad Dawud Ansari, and Pritha Gupta. Design and Implementation of Autonomous Car using Raspberry Pi. International Journal of Computer Applications. 113. 22-29.	2015	Aims to build an autonomous car prototype using Raspberry Pi as a processing chip.	In this paper, a method to make a self driving robot car is presented. The different hardware components and their assembly are clearly described.	Research Paper focuses more on Hardware rather than Machine learning algorithm to be used.	Gives a detailed idea of what hardware technologies will be required to build a prototype and also the functionality of each hardware part.
2.	Wu, Linxiu, Houjie Li, Jianjun He, and Xuan Chen. Traffic sign detection method based on Faster R-CNN. Journal of Physics: Conference Series. 1176. 032045. 10.1088/1742-6596/1176/3/032045.	2019	In this article, a traffic sign detection method comes up based on Faster R-CNN deep learning framework.	It is shown that this method is effective and robust to natural environments such as different lighting, occlusion, motion blur.	More focuses on complex natural environments such as different lighting, motion blur, sign occlusion and so on.	The traffic sign detection algorithm proposed to automatically extract features and realizes end-to-end training mode by using RPN.
3.	Ghosh, Rohan, Abhishek Mishra, Garrick Orchard, and Nitish V. Thakor. "Real-time object recognition and orientation estimation using an event-based camera and CNN," *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, Lausanne, 2014, pp.	2014	In this paper we describe a real-time bio-inspired system for object tracking and identification which combines an event-based vision sensor with a convolutional neural network running on FPGA for recognition.	Visual tracking and recognition of moving objects in cluttered scenes is typically regarded as a computaionally intensive task for artificial vision system.	This technique required intensive hardware to work on a prototype.	According to Research paper it work efficiently in practically.
4.	Thorat, Z.V., Mahadik, S., Mane, S., Mohite, S. and Udugade Self Driving Car using Raspberry-Pi and Machine Learning	2019	The paper aims to represent a monocular vision autonomous car prototype using the Raspberry-Pi	The different hardware components and their assembly	Research Paper focuses more on Hardware rather than	The Hardware functionality is clearly described.

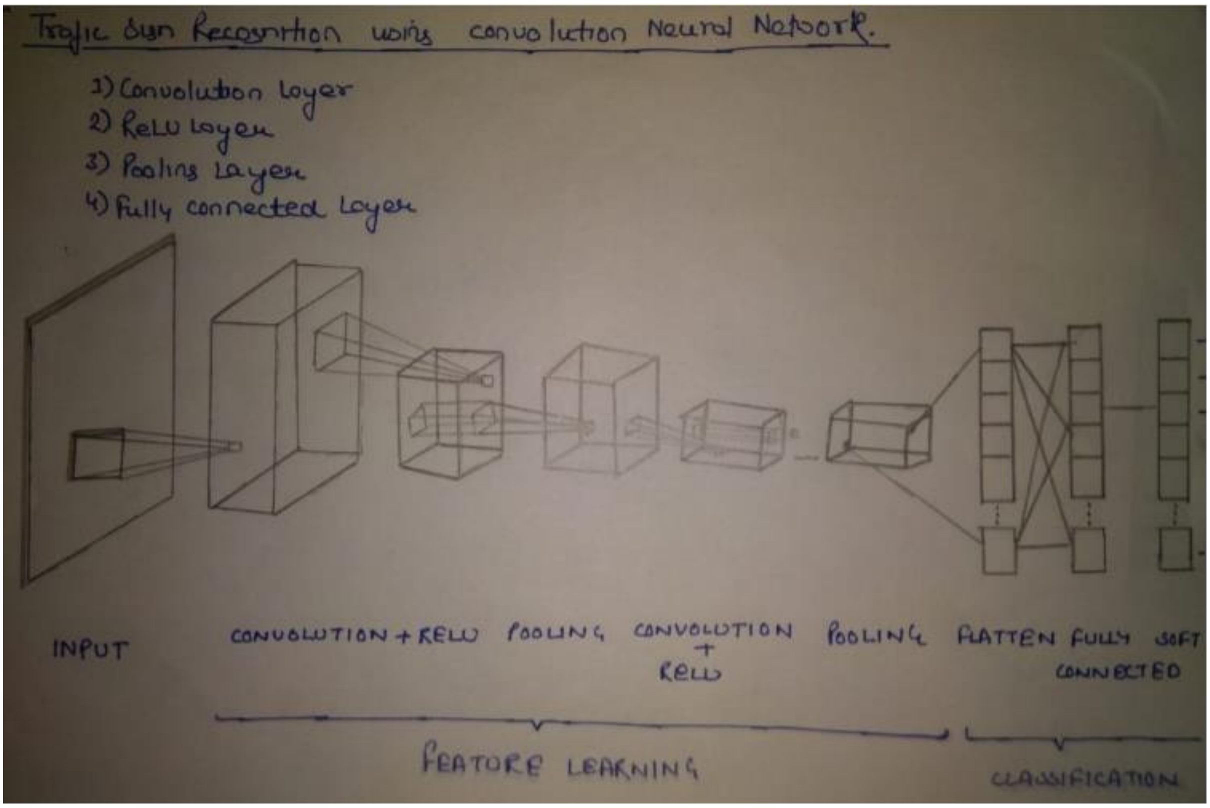
	International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 06 Issue: 03 | Mar 2019 www.irjet.net p-ISSN: 2395-0072		as a processing chip. The pi-camera module along with an ultrasonic sensor is used to provide necessary data from the real world to the car which would then pass the data on to the raspberry-pi.	are clearly described.	Machine learning algorithm to be used.	
5.	"Lu, Yifan, Jiaming Lu, Songhai Zhang, and Peter Hall. "Traffic signal detection and classification in street views using an attention model." Computational Visual Media 4, no. 3 (2018): 253-266."	2018	The attention model is designed to generate a small set of candidate regions at a suitable scale so that small targets can be better located and classified.	It presents a novel framework that utilizes a visual attention model to make detection more efficient, without loss of accuracy, and which generalizes.	It focuses on a special case: the detection and classification of traffic signals in street view.	Experiments show that this method has superior detection performance and is quicker than the general faster RCNN object detection framework on both datasets.
6.	W. Ye, S. Yuetian, X. Yunhe, W. Shu and Z. Yuchen, "The Implementation of Lane Detective Based on OpenCV," 2010 Second WRI Global Congress on Intelligent Systems, Wuhan, 2010, pp. 278-281, doi: 10.1109/GCIS.2010.120.	2010	This paper presents a lane detection algorithm that based on Hough transform. Principle of the algorithm and the implementation base on OpenCV are discussed in detail.	In intelligent vehicle systems, lane detection is one of the most important parts.		A research onthe lane line detection based on OpenCV is presented in this paper. Using LMed Square idea to select optimum subset in the lane line feature points so that eliminating unnecessary noise and improving the system's anti-noise ability.
7.	S. K. Vishwakarma, Akash and D. S. Yadav, "Analysis of lane detection techniques using openCV," 2015 Annual IEEE India Conference (INDICON), New Delhi, 2015, pp. 1-4, doi: 10.1109/INDICON.2015.7443166.	2015	It used Receiver Operating Characteristic curve (referred to as ROC hereafter) and Detection Error Trade-off curve (referred to as DET hereafter) which establish the accuracy of computer vision	Lane detection is one of the most challenging problems in machine vision. Lane detection faces all these challenges	Lane detection faces all these challenges due to loss of visibility, types of roads, road structure, road texture and other	The performance of two methods has been analyzed and compared using standard computer vision performance evaluation methods and it was found that method based on

			methods.	of loss of visibility, types of roads, road structure, road texture and other obstacles like trees, passing vehicles and their shadows.	obstacles like trees, passing vehicles and their shadows. There are several lane detection methods having their own working principles and backgrounds, merits and demerits.	Canny edge detection was better than the other one based on Sobel operator.
8.	Xiaodong Miao, Shunming Li, Huan Shen College of Energy and Power Nanjing University of Aeronautics and Astronautics Nanjing, China, 210016. ON-BOARD LANE DETECTION SYSTEM FOR INTELLIGENT VEHICLE BASED ON MONOCULAR VISION INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS, VOL. 5, NO. 4, DECEMBER 2012	2012	The objective of this research is to develop a monocular vision system that can locate the positions of the road lane in real time	The proposed system is shown to work well under various conditions on the roadway. Besides, the computation cost is inexpensive and the system's response is almost real time.	No assumptions are made about road structure, marking, or lane type, etc.	A five steps lane detection scheme that can successfully locate the lane line or boundary. In addition, it is also effective in various bad road scenes.
9.	Li, Qingquan & Chen, Long & Li, Ming & Shaw, Shih-Lung & Nuchter, Andreas. (2014). A Sensor-Fusion Drivable-Region and Lane-Detection System for Autonomous Vehicle Navigation in Challenging Road Scenarios. Vehicular Technology, IEEE Transactions on. 63. 540-555. 10.1109/TVT.2013.2281199.	2014	This paper presents a novel real-time optimal-drivable-region and lane detection system for autonomous driving based on the fusion of Light Detection and Ranging (LIDAR) and vision data.		Autonomous vehicle navigation is challenging since various types of road scenarios in real urban environments have to be considered, especially when only perception sensors are used, without position information.	A multi-level fusion-based road detection system for unmanned vehicle navigation in challenging road conditions, including structured roads, unstructured roads, and transitions between them, has been presented.

10.	Varghese, Jaycil Z., and Randy G. Boone. "Overview of autonomous vehicle sensors and systems." In *International Conference on Operations Excellence and Service Engineering*, pp. 178-191. 2015.	2015	This Paper will identify the application of various technologies that enable autonomous vehicles and also explain the advantages and disadvantages associated with each autonomous vehicle sensor.	Specific sensor and systems an show favorable results and can increase the efficiency of autonomous car.		We are presenting a short overview of the sensors and sensor fusion in autonomous vehicles. We focused on the sensor fusion from the key sensors in autonomous vehicles: camera, radar, and lidar .

Background Study and Findings

CNN image classifications take an input image, process it and classify it under certain categories. Each input image will pass it through a series of convolution layers with filters. Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

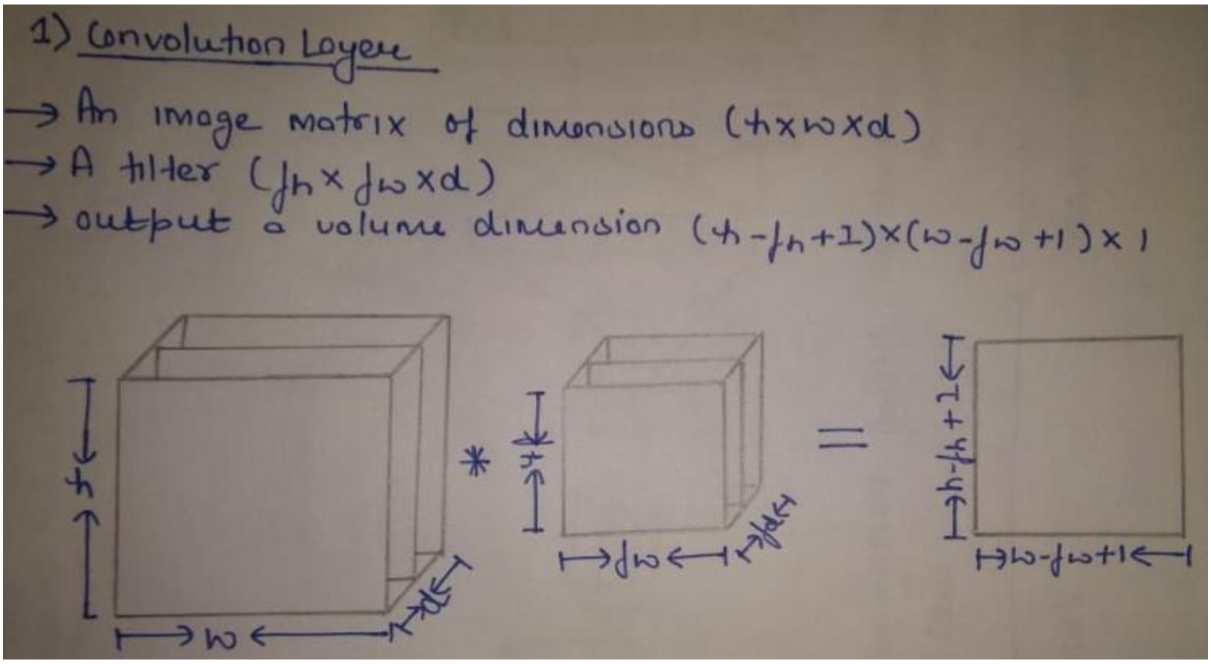


CNN have following layers –

1) Convolution Layer
2) ReLU Layer
3) Pooling Layer
4) Fully Connected Layer

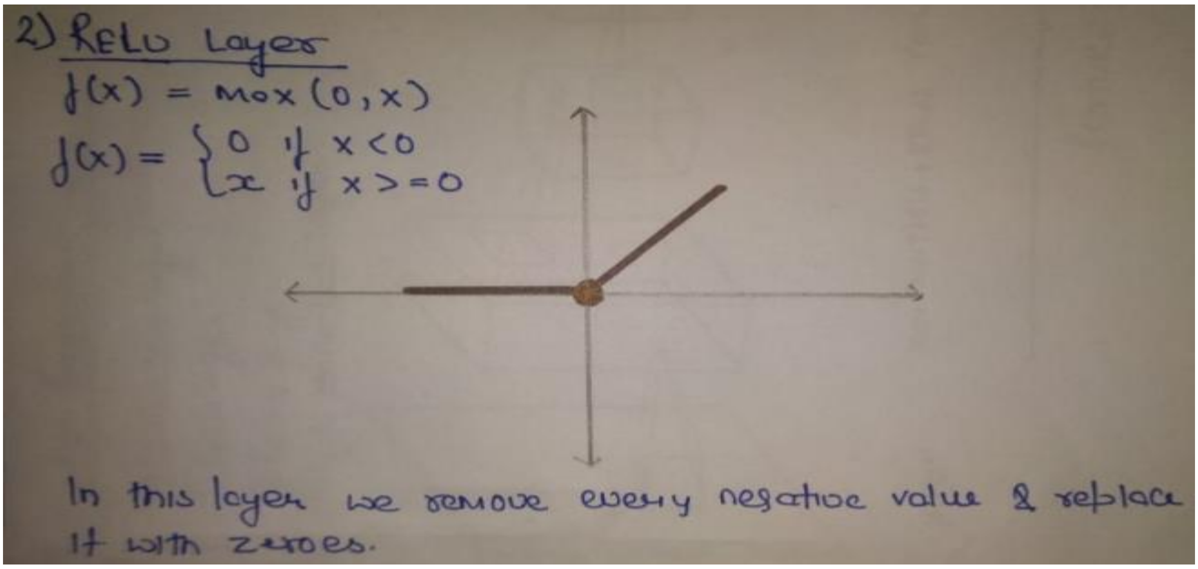
Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.



ReLU Layer

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is (x) = max(0,x).



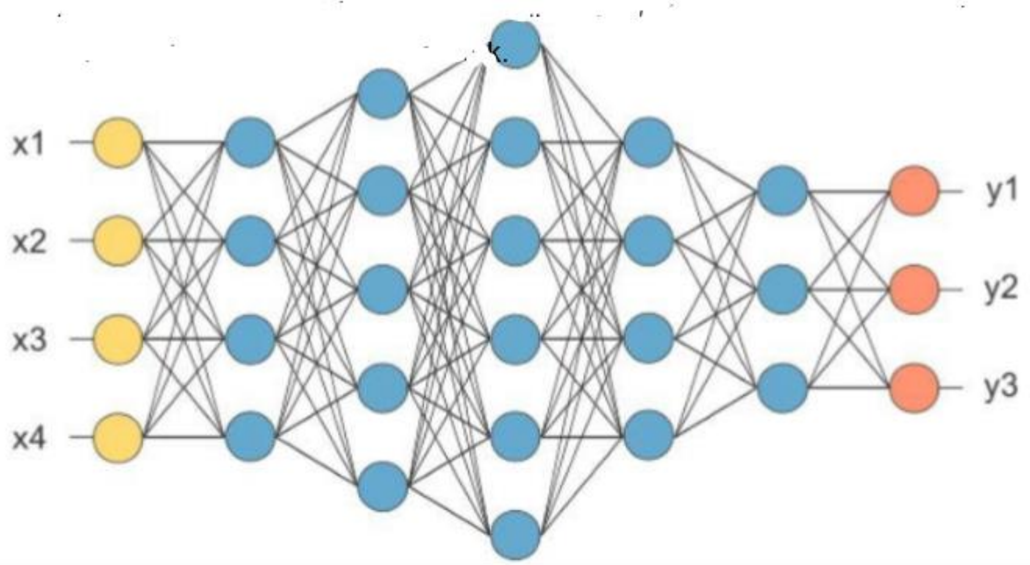
Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called sub sampling or down sampling which reduces the dimensionality of each map but retains important information.

Spatial pooling can be of different types:
Max pooling
Average pooling
Sum pooling

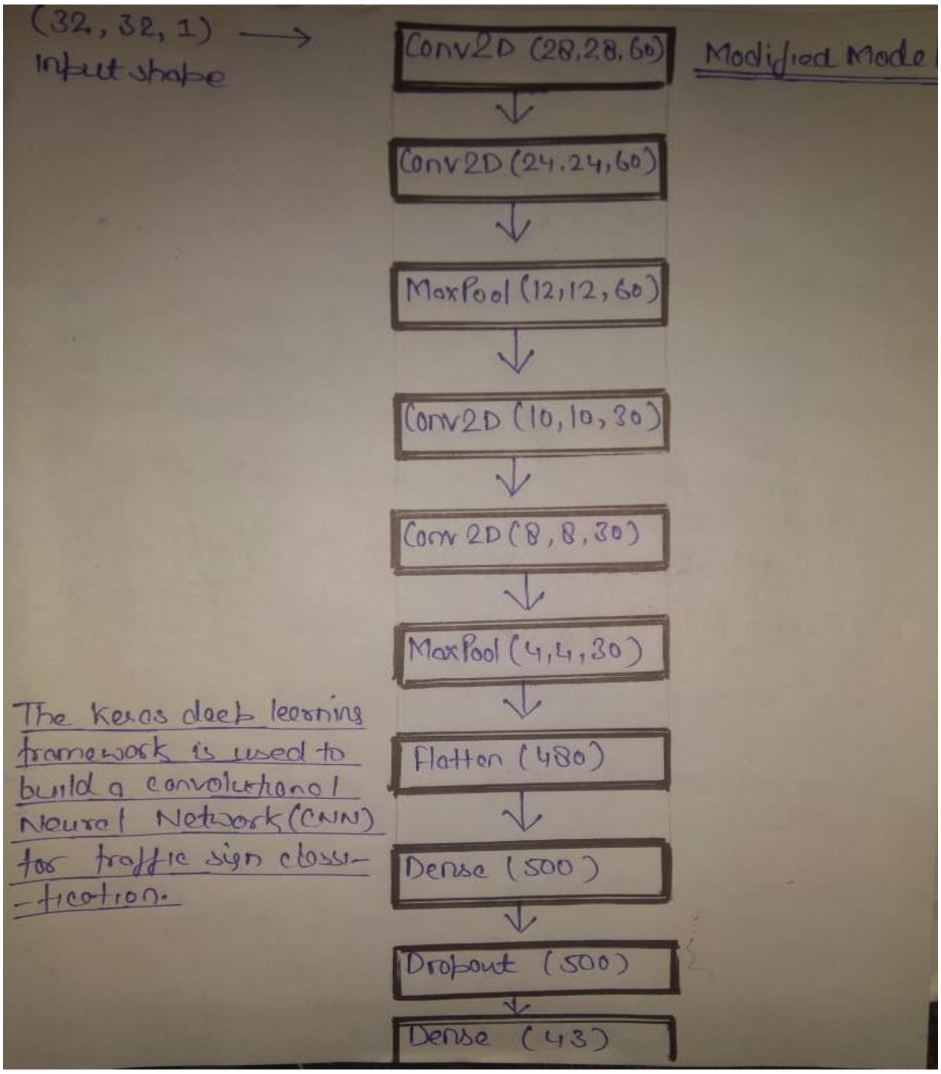
Fully Connected Layer

The layer we call the FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network



Algorithm

Classification with artificial neural networks is a very popular approach to solve pattern recognition problems. A neural network is a mathematical model based on connected via each other neural units artificial neurons similarly to biological neural networks. Typically, neurons are organized in layers, and the connections are established between neurons from only adjacent layers. The input low-level feature vector is put into first layer and, moving from layer to layer, is transformed to the high-level features vector. The output layer neurons amount is equal to the number of classifying classes. Thus, the output vector is the vector of probabilities showing the possibility that the input vector belongs to a corresponding class.
Each Convolutional layer consists of a set of trainable filters and computes dot productions between these filters and layer input to obtain an activation map. These filters are also known as kernels and allow detecting the same features in different locations.



Above Table describes our developed network architecture. The architecture consists of several Convolutional layers, fully connected layers, softmax layer. All Convolutional layers have parameter stride equal to 1. This parameter determines the stride of the convolution sliding window, so layers with parameter stride greater than 1 also combine the pooling operation. The softmax layer normalizes the previous layer output so that its output contains probabilities of belonging to recognizable classes for the original input image.

Traffic Sign Recognition

Preprocessing of Data

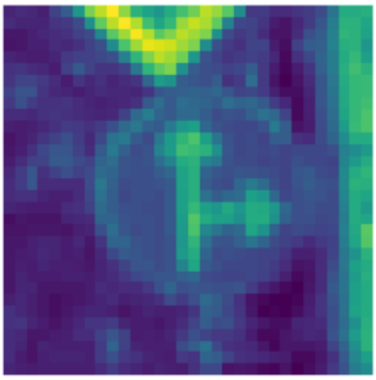
```
[ ]
import cv2

plt.imshow(X_train[1000])
plt.axis("off")
print(X_train[1000].shape)
print(y_train[1000])
def grayscale(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img
```

```
(32, 32, 3)
36
```

```
[ ]
img = grayscale(X_train[1000])
plt.imshow(img)
plt.axis("off")
print(img.shape)
```

```
(32, 32)
```



```
def equalize(img):
    img = cv2.equalizeHist(img)
    return img
img = equalize(img)
plt.imshow(img)
plt.axis("off")
print(img.shape)
```

```
[ ]
def preprocess(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img
```

Preprocessing train, test and validation datasets

```
[ ]

    X_train = np.array(list(map(preprocess, X_train)))
    X_test = np.array(list(map(preprocess, X_test)))
    X_val = np.array(list(map(preprocess, X_val)))
```

Reshaping images to 32*32*1

```
X_train = X_train.reshape(34799, 32, 32, 1)
X_test = X_test.reshape(12630, 32, 32, 1)
X_val = X_val.reshape(4410, 32, 32, 1)
```

Image Data Generator

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(width_shift_range=0.1,
                            height_shift_range=0.1,
                            zoom_range=0.2,
                            shear_range=0.1,
                            rotation_range=10.)

datagen.fit(X_train)
# for X_batch, y_batch in


batches = datagen.flow(X_train, y_train, batch_size = 15)
X_batch, y_batch = next(batches)

fig, axs = plt.subplots(1, 15, figsize=(20, 5))
fig.tight_layout()

for i in range(15):
    axs[i].imshow(X_batch[i].reshape(32, 32))
    axs[i].axis("off")

print(X_batch.shape)
```

Modified Model of CNN

```
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)
y_val = to_categorical(y_val, 43)

# create model

def modified_model():
  model = Sequential()
  model.add(Conv2D(60, (5, 5), input_shape=(32, 32, 1), activation='relu'))
  model.add(Conv2D(60, (5, 5), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))

  model.add(Conv2D(30, (3, 3), activation='relu'))
  model.add(Conv2D(30, (3, 3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))

  model.add(Flatten())
  model.add(Dense(500, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(43, activation='softmax'))

  model.compile(Adam(lr = 0.001), loss='categorical_crossentropy', metrics=['accuracy'])
  return model
model = modified_model()
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 60)        1560
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 60)        90060
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 60)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 30)        16230
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 30)          8130
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 30)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 480)               0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               240500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 43)                21543
=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
_________________________________________________________________
None
```

Training the Model

```
[ ]

    history = model.fit_generator(datagen.flow(X_train, y_train, batch_size=50),
                                  steps_per_epoch=2000,
                                  epochs=10,
                                  validation_data=(X_val, y_val), shuffle = 1)
```

No of Epochs = 10

Steps per Epochs = 2000

```
Epoch 1/10
2000/2000 [==============================] - 35s 18ms/step - loss: 0.8968 - accuracy: 0.7398 - val_loss: 0.0971 - val_accuracy: 0.9710
Epoch 2/10
2000/2000 [==============================] - 35s 17ms/step - loss: 0.2162 - accuracy: 0.9331 - val_loss: 0.0580 - val_accuracy: 0.9864
Epoch 3/10
2000/2000 [==============================] - 34s 17ms/step - loss: 0.1484 - accuracy: 0.9545 - val_loss: 0.0422 - val_accuracy: 0.9880
Epoch 4/10
2000/2000 [==============================] - 35s 17ms/step - loss: 0.1181 - accuracy: 0.9629 - val_loss: 0.0313 - val_accuracy: 0.9898
Epoch 5/10
2000/2000 [==============================] - 35s 17ms/step - loss: 0.0991 - accuracy: 0.9696 - val_loss: 0.0412 - val_accuracy: 0.9882
Epoch 6/10
2000/2000 [==============================] - 34s 17ms/step - loss: 0.0882 - accuracy: 0.9725 - val_loss: 0.0543 - val_accuracy: 0.9896
Epoch 7/10
2000/2000 [==============================] - 34s 17ms/step - loss: 0.0777 - accuracy: 0.9760 - val_loss: 0.0346 - val_accuracy: 0.9884
Epoch 8/10
2000/2000 [==============================] - 34s 17ms/step - loss: 0.0694 - accuracy: 0.9789 - val_loss: 0.0265 - val_accuracy: 0.9932
Epoch 9/10
2000/2000 [==============================] - 35s 17ms/step - loss: 0.0649 - accuracy: 0.9806 - val_loss: 0.0264 - val_accuracy: 0.9909
Epoch 10/10
2000/2000 [==============================] - 34s 17ms/step - loss: 0.0627 - accuracy: 0.9816 - val_loss: 0.0517 - val_accuracy: 0.9882
```

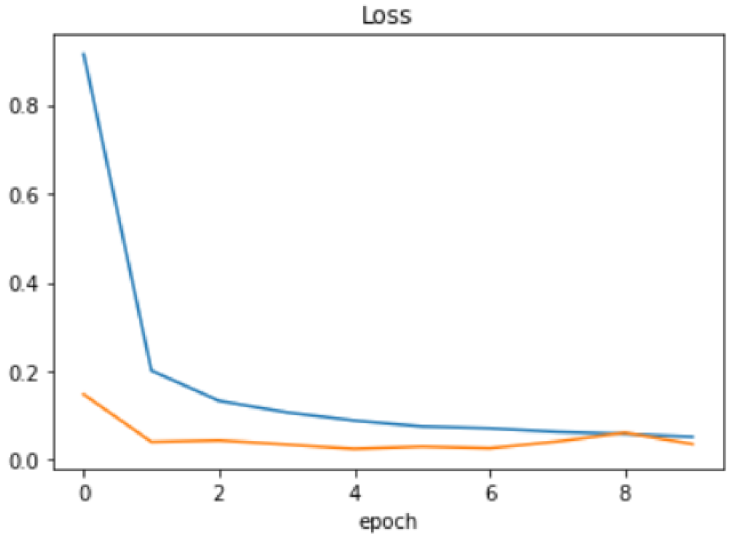
Val_accuracy = 0.9882

Val_loss = 0.0517

Graph of Loss and Accuracy

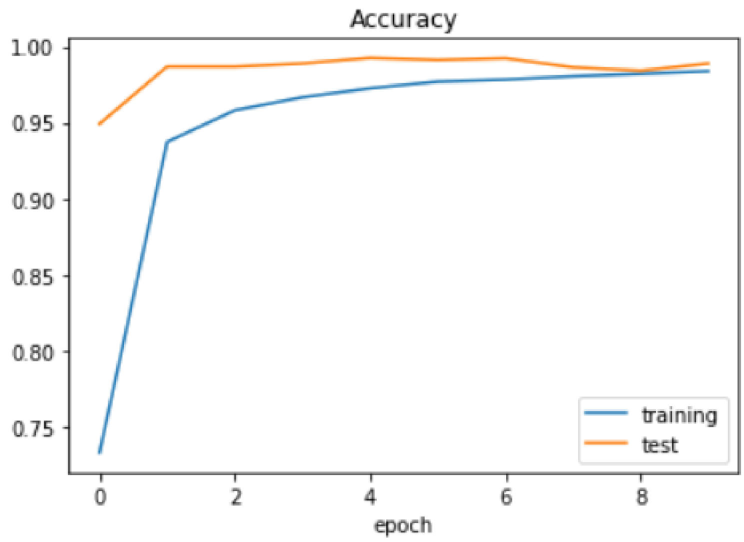
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss')
plt.xlabel('epoch')
```

```
Text(0.5, 0, 'epoch')
```



```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','test'])
plt.title('Accuracy')
plt.xlabel('epoch')
```

```
Text(0.5, 0, 'epoch')
```



Accuracy of Test dataset

```
[ ]
# TODO: Evaluate model on test data
score = model.evaluate(X_test, y_test, verbose=0)

print('Test score:', score[0])
print('Test accuracy:', score[1])
```

```
Test score: 0.11607063797850332
Test accuracy: 0.973792552947998
```

Accuracy = 97.37%

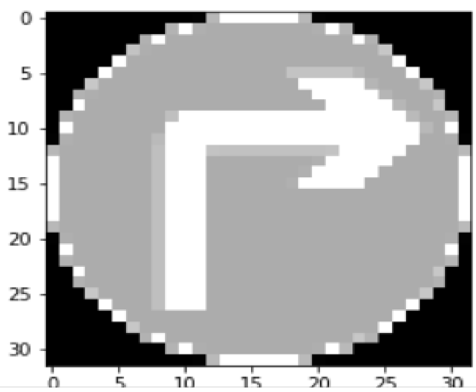
Input of an image -

```
#predict internet number
import requests
from PIL import Image
url = 'https://traffic-rules.com/img/asia/in/signs/mandatory/mandatory-direction-ahead-turn-right.png'
r = requests.get(url, stream=True)
img = Image.open(r.raw)
plt.imshow(img, cmap=plt.get_cmap('gray'))

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)
s = str(model.predict_classes(img))
print("predicted sign: "+ name_of_class[int(s[1:(len(s)-1)])])
```

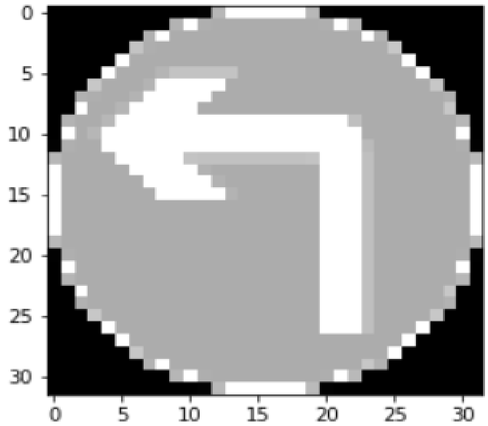
Output

```
(32, 32)
predicted sign: 33 - Turn right ahead
```



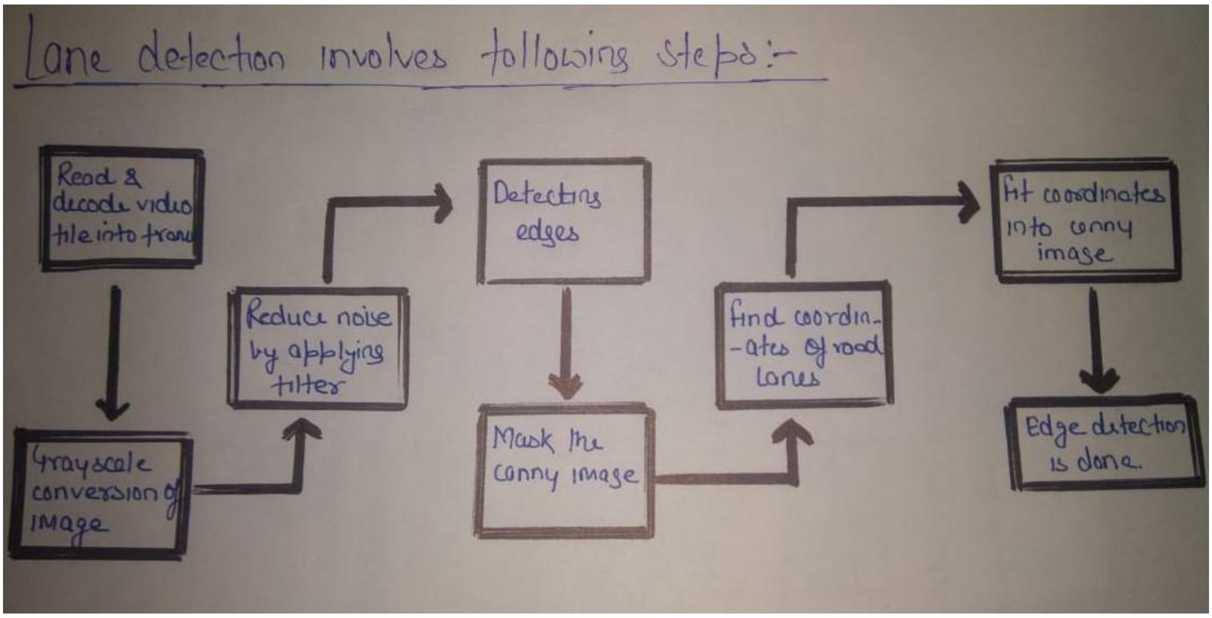
ve.google.com/drive/search?q=owner%3Ame %28type%3Aapplication%2Fvnd.goo...

```
(32, 32)
predicted sign: 34 - Turn left ahead
```



Lane detection using opencv

Autonomous Driving Car is one of the most disruptive innovations in AI. Fuelled by Deep Learning algorithms, they are continuously driving our society forward and creating new opportunities in the mobility sector. An autonomous car can go anywhere a traditional car can go and does everything that an experienced human driver does. But it’s very essential to train it properly. One of the many steps involved during the training of an autonomous driving car is lane detection.



- Decoding video file**:** After the capturing has been initialized every video frame is decoded (i.e. converting into a sequence of images).

- 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.

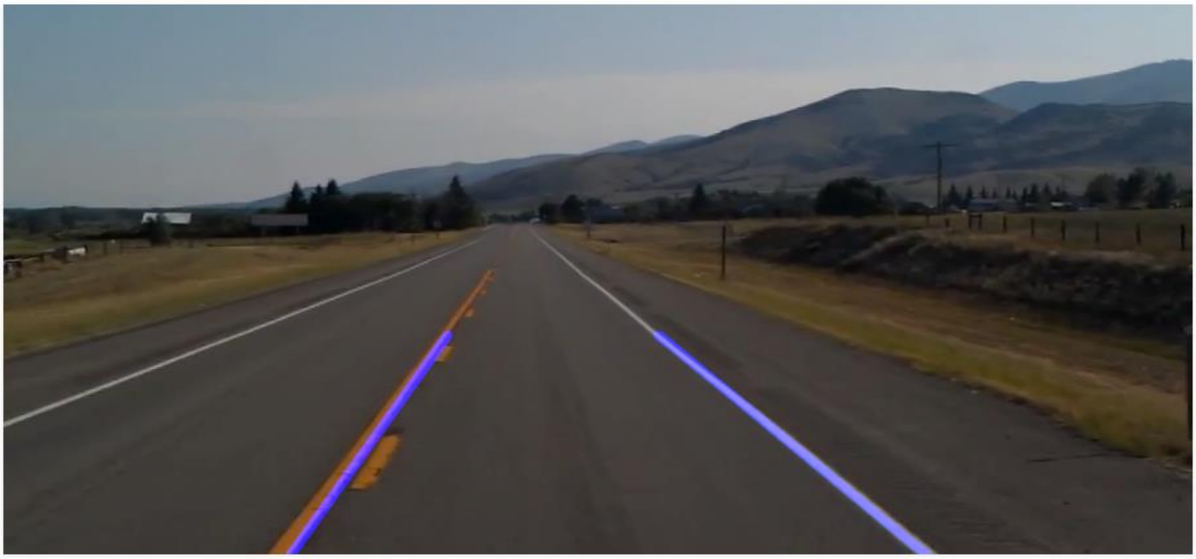
- Reduce noise**:** Noise can create false edges, therefore before going further, it's imperative to perform image smoothening. Gaussian filter is used to perform this process.
- Canny Edge Detector: It computes gradient in all directions of our blurred image and traces the edges with large changes in 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.

Input :

Output :



Instead of letting Convolutional neural network choose its kernel, some predefined kernels used for image processing such as sharpening, edge-detecting, discrete cosine transformation and blurring were applied to the first layer of CNN. This concept was named as the General Filter Convolutional Neural Network (GFNN).
The results for this model GFCNN showed 30% less training time than that of CNN and the accuracy was higher compared to CNN despite the model being quicker.

Under the architecture, as the processing happens on each layer of convolutional neural network, the target area in the image becomes smaller.

The Canny edge detection technique

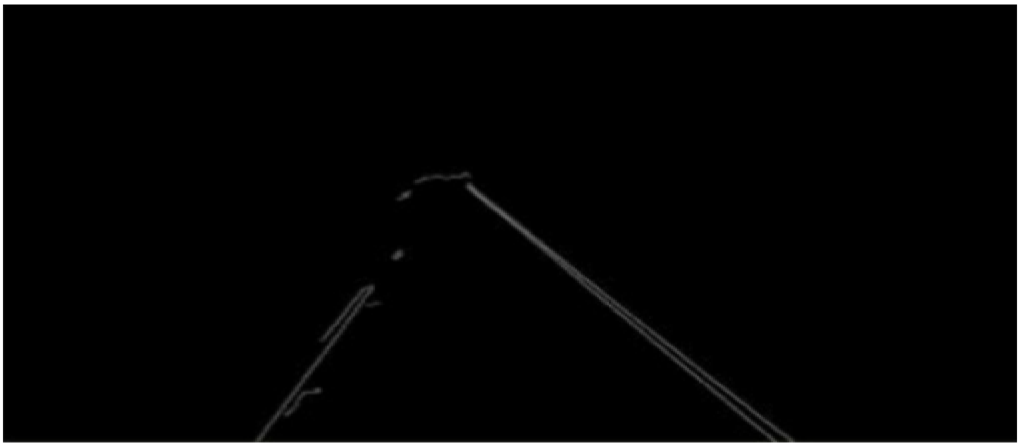
The Canny Detector is a multi-stage algorithm optimized for fast real-time edge detection. The fundamental goal of the algorithm is to detect sharp changes in luminosity (large gradients), such as a shift from white to black, and defines them as edges, given a set of thresholds.
An image is an array of pixel and pixel store data about intensity ranging from 0 to 255. Gradient denotes the change in brightness in a series of pixels.

Gaussian Blur

We have to smooth the image before processing it. It is done by modifying the value of the pixel by the average value of the pixel around it.

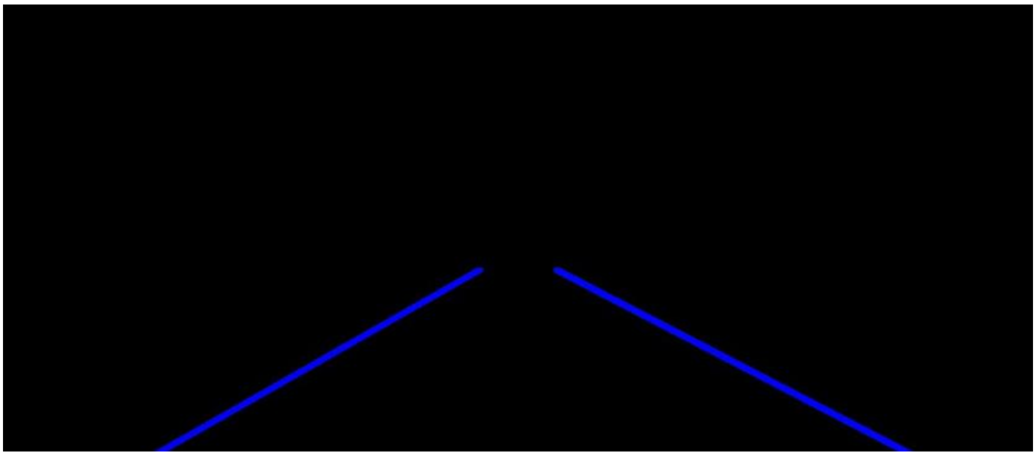
Edge Detection

An edge corresponds to the region in an array where there is sharp change in the intensity or color between adjacent pixels.

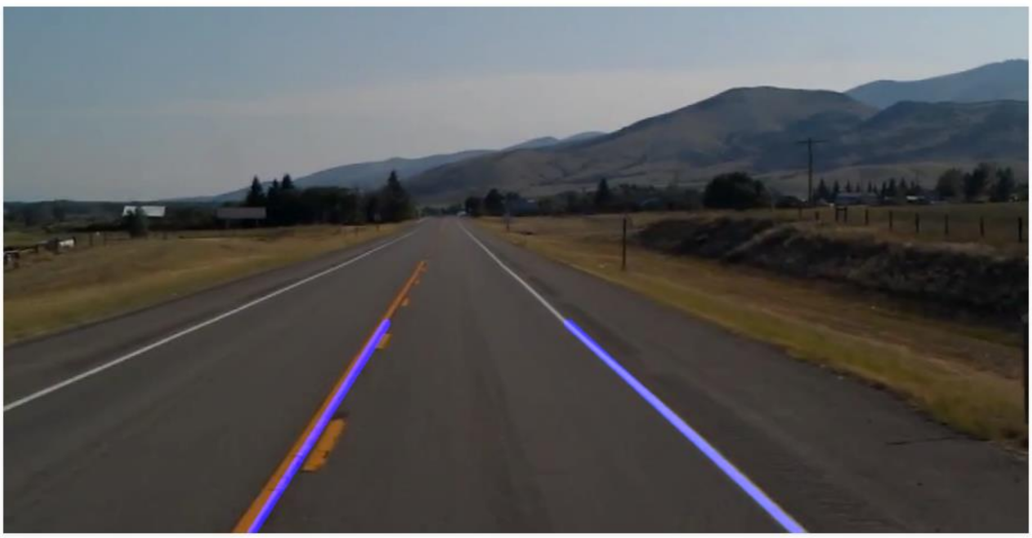


Hough Transform

Now we make use of Hough transform technique that will detect straight lines in the image and thus identify the lane lines.

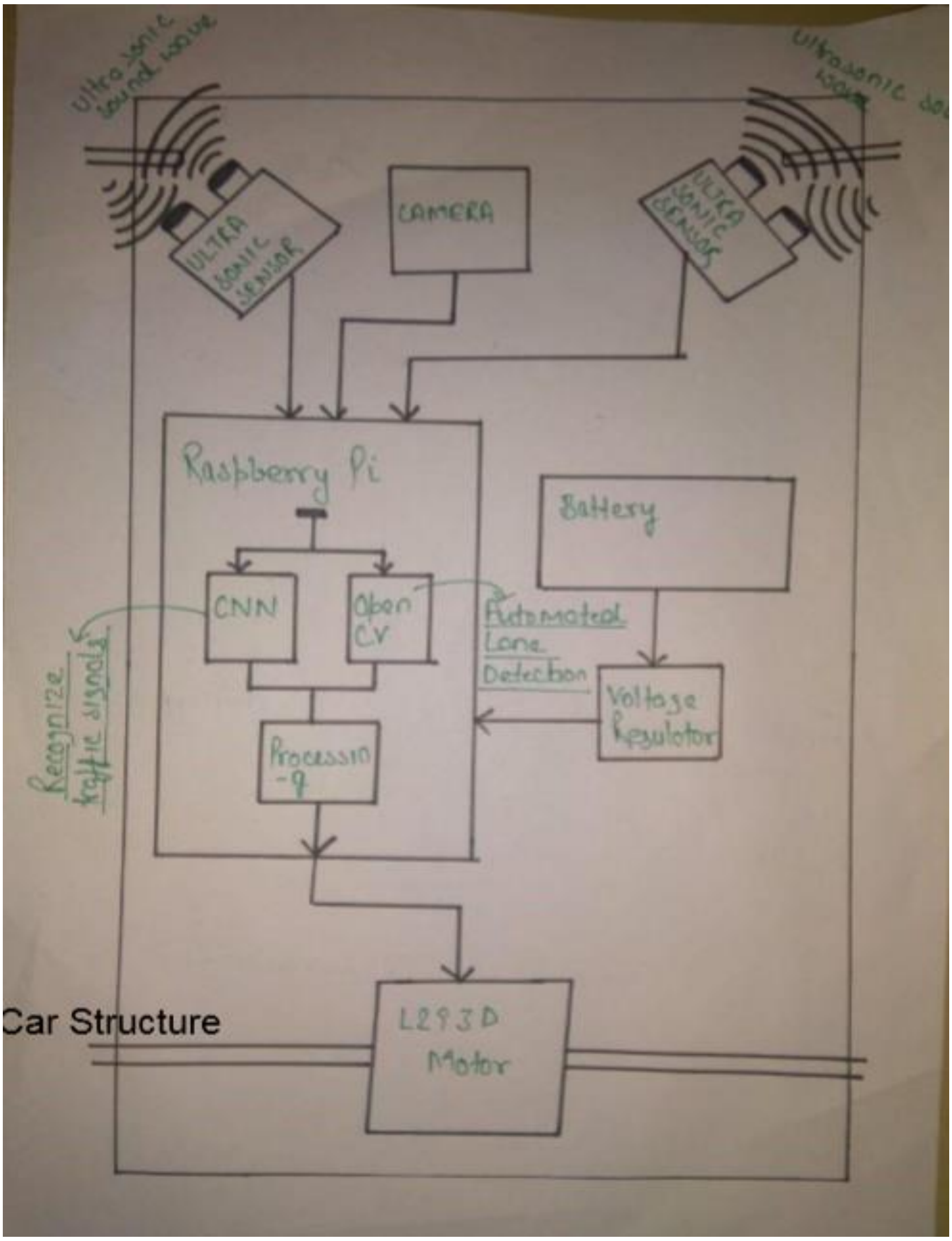


Output :



Conclusion

An autonomous car can be built with some extra hardware an, Traffic recognition system and Automatic Lane detection system integrated within Raspberry pi. Below is the Structure of a simple autonomous car.



References

1. Pannu, Gurjashan Singh, Mohammad Dawud Ansari, and Pritha Gupta. Design and Implementation of Autonomous Car using Raspberry Pi. International Journal of Computer Applications. 113. 22-29.

2. Wu, Linxiu, Houjie Li, Jianjun He, and Xuan Chen. Traffic sign detection method based on Faster R-CNN. Journal of Physics: Conference Series. 1176. 032045. 10.1088/1742-6596/1176/3/032045.

3. Ghosh, Rohan, Abhishek Mishra, Garrick Orchard, and Nitish V. Thakor.Real-time object recognition and orientation estimation using an event-based camera and CNN. In *2014 IEEE Biomedical Circuits and Systems Conference Proceedings* (pp. 544-547). IEEE.

4. Thorat, Z.V., Mahadik, S., Mane, S., Mohite, S. and Udugade, ASelf Driving Car using Raspberry-Pi and Machine Learning International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 06 Issue: 03 | Mar 2019 p-ISSN: 2395-0072

5. "Lu, Yifan, Jiaming Lu, Songhai Zhang, and Peter Hall. "Traffic signal detection and classification in street views using an attention model." Computational Visual Media 4, no. 3 (2018): 253-266."

6. W. Ye, S. Yuetian, X. Yunhe, W. Shu and Z. Yuchen, "The Implementation of Lane Detective Based on OpenCV," 2010 Second WRI Global Congress on Intelligent Systems, Wuhan, 2010, pp. 278-281, doi: 10.1109/GCIS.2010.120.

7. S. K. Vishwakarma, Akash and D. S. Yadav, "Analysis of lane detection techniques using openCV," 2015 Annual IEEE India Conference (INDICON), New Delhi, 2015, pp. 1-4, doi: 10.1109/INDICON.2015.7443166.

8. ON-BOARD LANE DETECTION SYSTEM FOR INTELLIGENT VEHICLE BASED ON MONOCULAR VISION Xiaodong Miao, Shunming Li, Huan Shen College of Energy and Power Nanjing University of Aeronautics and Astronautics Nanjing, China, 210016. INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS, VOL. 5, NO. 4, DECEMBER 2012

9. Li, Qingquan & Chen, Long & Li, Ming & Shaw, Shih-Lung & Nuchter, Andreas. (2014). A Sensor-Fusion Drivable-Region and Lane-Detection System for Autonomous Vehicle Navigation in Challenging Road Scenarios. Vehicular Technology, IEEE Transactions on. 63. 540-555. 10.1109/TVT.2013.2281199.

10. Varghese, Jaycil Z., and Randy G. Boone. "Overview of autonomous vehicle sensors and systems." In *International Conference on Operations Excellence and Service Engineering*, pp. 178-191. 2015.