Annexure-1 (A typical Specimen of TitlePage)

TRAFFIC SIGNAL CLASSFICATION USING LE-NET

A Project Work Synopsis

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE &

MACHINE LEARNING

Submitted by:

NAME OF THE STUDENT

Tushar Mahajan – 18BCS6105

Madhav Mundhra – 18BCS6107

Nihal Agarwalla – 18BCS6106

Aman Kothari -18BCS6097

Under the Supervision of:

Gurpreet Singh

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

APRIL 2021

Annexure-2

DECLARATION

We, students of **'Bachelor of Engineering in Artificial Intelligence & Machine Learning**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled '**Traffic Signal Classification using Le-Net**' is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Tushar Mahajan (6105)
Madhav Mundhra (6107)
Nihal Agarwalla (6106)
Aman Kothari (6097)

Date:
April, 2021

Place:
Chandigarh University, Mohali

Annexure-3 **(A typical specimen of table of contents)**

Table of Contents

List of Tables

.

List of Figures

1 INTRODUCTION

In the modern era of civilization detection of traffic signals for autonomous cars have become an important factor for compiling with the standards of traffic disciplines which are imposed by the governments for safer and efficient movements of vehicles. Traffic congestion and accidents are major issues. One of the chief difficulties in rush hour gridlock control is to oblige the traffic in a sheltered and productive manner. At the point when traffic request is extraordinary enough that the collaboration between vehicles eases back the speed of the traffic stream, this outcomes in some clog.

Our proposed model aims to analyze the current traffic situations on the road, dangers and difficulties near the vehicle, alerting the passengers in case of any threats, and provide a convenient and healthier navigation containing all the necessary information. The model provides the necessary information to the self-driving car or to the passenger which results in the car taking necessary actions in case of any threats or difficulties. We use the technique of le net to classify all the signs and symbols detected by our sensors.

Our sensors have the ability to detect the images of sizes as low as 32*32 pixels so that our autonomous vehicle doesn't miss any signs irrespective of the quality of the image being used. The Le-Net architecture is an excellent "first architecture" for Convolutional Neural. Like pretty much every other neural system they are prepared with an adaptation of the explicitly planned calculations. They can perceive designs with outrageous inconstancy and with heartiness to twists and straightforward geometric changes.

1.1 Deep Learning:

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled, also known as deep neural learning or deep neural network.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing. However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unravelling

this wealth of information and are increasingly adapting to AI systems for automated support. Deep learning learns from vast amounts of unstructured data that would normally take humans decades to understand and process.

1.2 Neural Networks

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus, a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modelled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be 1 and 1.

These artificial networks may be used for predictive modelling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information. Artificial intelligence, cognitive modelling, and neural networks are information processing paradigms inspired by the way biological neural systems process data. Artificial intelligence and cognitive modelling try to simulate some properties of biological neural networks. In the artificial intelligence field, artificial neural networks have been applied successfully to speech recognition, image analysis and adaptive control, in order to construct software agents (in computer and video games) or autonomous robots.

Historically, digital computers evolved from the von Neumann model, and operate via the execution of explicit instructions via access to memory by a number of processors. On the other hand, the origins of neural networks are based on efforts to model information processing in biological systems. Unlike the von Neumann model, neural network computing does not separate memory and processing. Neural network theory has served both to better identify how the neurons in the brain function and to provide the basis for efforts to create artificial intelligence.

2 LITERATURE REVIEW

Title	Year of Publication	Authors	Description	Result
Automatic detection of traffic lights changes from red to green and car turn signals in order to improve urban traffic	**2014**	**Julian Balcerek, Adam Konieczka ; Tomasz Marcinia, Adam Dabrowski, Krzysztof Maćkowiak, Karol Piniarski**	**The purpose of the paper was detection of traffic lights changes and vehicle indicator signals in order to improve urban traffic In this paper two vision systems for urban traffic improvement are presented. In both of them information from a camera mounted on a vehicle is used. The first system automatically detects traffic lights changes from red to green and allows to start movement of the car without unnecessary delays. The second system automatically detects turn signal of a car, which is going to change the lane.**	**This provides information to help the proper use of the late merge scheme in situations when two lanes in one direction are merged into one lane. Initial experiments indicate high efficiency of the proposed mechanisms**
Image Processing Based Traffic Sign Detection and Recognition with Fuzzy Integral	**2016**	**Canan Taştimur, Mehmet Karaköse, Yavuz Çelik, Erhan Akın**	**The purpose of this paper was to build a system that can automatically recognize the traffic signs to reduce traffic accidents. This study includes traffic sign detection and recognition application. In this study, image processing techniques and Fuzzy Integral is used to recognize traffic signs. The proposed method consists of 3 stages. These stages are Image Acquisition, Traffic Sign Detection and**	**An algorithm based contactless image processing using Fuzzy Integral was suggested to detect and to recognize the traffic sign detection. Traffic sign was detected using traffic signs imaged under diverse angle and lightning cases and Regulatory, information, Warning and Prohibitory signs were recognized with**

			Traffic Sign Recognition	**the proposed method.**
An Efficient Framework for Recognizing Traffic lights in Night Traffic Images	**2012**	**Bo Fan, Weiyao Lin, Xiaokang Yang**	**In this paper, the authors propose a new and efficient framework for detecting and recognizing traffic signal lights with the corresponding countdown counters in the night scenes. The proposed framework first identifies the possible objects with the residuals of color selection. Then the shooting angles are corrected for obtaining the frontal direction image. Finally, the light signals are recognized based on templates matching measure.**	**Thus, a new and efficient framework is proposed for detecting and recognition of the traffic signal lights in the night situation. We can see that the correction method has high accuracy regardless of the value of shooting angles (wide or small angle).**
A gradient-domain image enhancement method for traffic signs in nighttime surveillance	**2017**	**Simin Wang, Li Zhuo, Hui Zhang, Xiaoguang Li**	**Simin Wang, Li Zhuo, Hui Zhang, Xiaoguang Li**	**The results is the average value of each descriptor. The method is characterized by positive value of the indicator 1 γ for the considered images, and 2 γ is the best in three methods.**
Real-time image processing approach to measure traffic queue parameters	**1995**	**M. Fathy, M.Y. Siyal**	**Measurement of various traffic parameters including queue parameters. Queue detection algorithm which is divided into spatial domain and frequency domain techniques.**	**It measures the queue parameters such as the period of occurrence between queues, the length and the slope of occurrence.**

2.1 EXISTING SYSTEM

In maximum Python project example, they all have built a deep neural network model that can classify traffic signs present in the image into different categories. With this model, we are able to read and understand traffic signs which are a very important task for all autonomous vehicles. According to the traffic signal light, the car has to perform. Maximum model were about to changing the approaches for the efficiency of the models.

2.2 PROBLEM STATEMENT

1. It will be difficult to classify, if the images or current scenario is dark.

2. More than 5 signs come between 50 meters so, performance may be varying.

3. Blur images can decrease our model performance.

4. Traffic signs should be at front 180 degrees of the car for the best performance.

5. Latest new traffic sign can mislead our mode

3 PROBLEM FORMULATION

We separate pictures into envelopes and give them their proper names, i.e. the preparation set and the test set. This makes it simpler to bring the pictures into Keras.

1) Setup: In this progression we have to import Keras and different bundles that were going to use in building the CNN. There are different bundles which are incorporated inside it: -

- Sequential is utilized to introduce the neural system.
- Convolution2D is utilized to make the convolutional organize that manages the pictures.
- MaxPooling2D layer is utilized to include the pooling layers.
- Flatten is the capacity that changes over the pooled include guide to a solitary section that is passed to the completely associated layer.
- Dense adds the completely associated layer to the neural system.

2) Initializing the neural system: To instate the neural system we make an object of the Sequential class.

3) Convolution: To include the convolution layer, we call the include work with the classifier article and go in Convolution2D with parameters. The principal contention nbfilter. nbfilter is the quantity of highlight indicators that we need to make. The second and third parameters are measurements of the component indicator grid. It's basic practice to begin with 32 element indicators for CNNs. The following parameter is input shape which is the state of the information picture. The pictures will be changed over into this shape during pre-handling. On the off chance that the picture is highly contrasting it will be changed over into a 2D exhibit and if the picture is hued it will be changed over into a 3D cluster.

4) Pooling: Right now decline the size of the component map. Generally we make a pool size of 2x2 for max pooling. This engages us to diminish the size of the component map while not losing noteworthy picture information.

5) Flattening: In this progression, all the pooled include maps are taken and placed into a solitary vector. The Flatten work straightens all the element maps into a solitary section.

6) Full association: The accompanying stage is to use the vector we got above as the commitment for the neural framework by using the Dense limit in Keras. The principal parameter is output dim which is the quantity of hubs in the shrouded layer. You can decide the most proper number

through experimentation. The higher the quantity of measurements the all the more figuring assets you should fit the model. A typical practice is to pick the quantity of hubs in forces of two. The subsequent parameter is the actuation work. We for the most part utilize the ReLu actuation work in the concealed layer. The following layer we need to include is the yield layer. Right now, utilize the sigmoid enactment work since we anticipate a paired result. In the event that we expected multiple results we would utilize the softmax work. The output_dim here is 1 since we simply anticipate the anticipated probabilities of the classes.

7) Compiling CNN: We at that point accumulate the CNN utilizing the aggregate capacity. This capacity anticipates three parameters: the streamlining agent, the misfortune work, and the measurements of execution. The enhancer is the angle plummet calculation we are going to utilize. We utilize the binary_crossentropy misfortune work since we are doing a double order.

8) Fitting the CNN: We are going to pre-process the pictures utilizing Keras to forestall over fitting. This preparing is known as picture increase. The Keras utility we use for this design is ImageDataGenerator. This capacity works by flipping, rescaling, zooming, and shearing the pictures. The principal contention rescale guarantees the pictures are rescaled to have pixel esteems somewhere in the range of zero and one. horizontal_flip=True implies that the pictures will be flipped on a level plane. Every one of these activities are a piece of the picture expansion.

9) Making a solitary forecast: Now that the model is fitted, we can utilize the foresee technique to make expectations utilizing new pictures. So as to do this we have to pre-process our pictures before we pass them to the anticipate strategy. To accomplish this, we'll utilize a few capacities from numpy. We likewise need to import the picture module from Keras to permit us to stack in the new pictures. The following stage is to stack the picture that we might want to foresee. To achieve this we utilize the load_img work from the picture module. The main contention this capacity takes is the way to the area of the picture and the subsequent contention is the size of the picture. The size of the picture ought to be equivalent to the size utilized during the preparation procedure.

4 RESEARCH OBJECTIVES

The proposed research is aimed to carry out work leading to the development of an approach for ..... The proposed aim will be achieved by dividing the work into following objectives:

1. Give a picture as contribution for the test.
2. Named the dataset.
3. Pre-handling, trimming, resizing.
4. Picture Augmentation.
5. Take information in a rundown position.
6. Taking Le-Net model of CNN design
7. CNN preparing
8. Information Image resized 32*32
9. CNN classifier

Picture Augmentation

In picture increase, because of absence of legitimate marked dataset for the preparation set for the model, that is the reason we are utilizing expansion for amplifying the preparation dataset for the model. Right now, a solitary picture will be changed over into 6 additional pictures by pivoting, invertation, making it inverse. Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. Training deep learning neural network models on more data can result in more skilful models, and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images. The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class. Which help dataset to change over into enormous dataset which will help in more approval accuracy.

The intent is to expand the training dataset with new, plausible examples. This means, variations of the training set images that are likely to be seen by the model. For example, a horizontal flip of a picture of a cat may make sense, because the photo could have been taken from

the left or right. A vertical flip of the photo of a cat does not make sense and would probably not be appropriate given that the model is very unlikely to see a photo of an upside down cat. As such, it is clear that the choice of the specific data augmentation techniques used for a training dataset must be chosen carefully and within the context of the training dataset and knowledge of the problem domain. In addition, it can be useful to experiment with data augmentation methods in isolation and in concert to see if they result in a measurable improvement to model performance, perhaps with a small prototype dataset, model, and training run.

Modern deep learning algorithms, such as the convolutional neural network, or CNN, can learn features that are invariant to their location in the image. Nevertheless, augmentation can further aid in this transform invariant approach to learning and can aid the model in learning features that are also invariant to transforms such as left-to-right to top-to-bottom ordering, light levels in photographs, and more. Image data augmentation is typically only applied to the training dataset, and not to the validation or test dataset. This is different from data preparation such as image resizing and pixel scaling; they must be performed consistently across all datasets that interact with the model.

Pre-processing

In our model, 2000 pictures marked dataset are utilized, where full picture isn't required to play the activity of the recognition thus, that time the pre-processing have the spot. It will permit the pictures to go for the trainable parameters. Image pre-processing is the name for operations on images at the lowest level of abstraction whose aim is an improvement of the image data that suppress undesired distortions or enhances some image features important for further processing. It does not increase image information content. Pre-processing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. These iconic images are of the same kind as the original data captured by the sensor, with an intensity image usually represented by a matrix of image function values (brightness). The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, translation) are classified among pre-processing methods here since similar techniques are used..Also, progressively trainable parameters prompt more exactness in the model and Le-net utilize 32*32 pixel pictures required for the information by editing and resizing. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing. Mage pre-processing methods use the considerable redundancy in images. Neighbouring pixels corresponding to one object in real images have essentially the same or similar brightness value. Thus, distorted pixel can often be restored as an average value of neighbouring pixels.

5 METHODOLOGY

We separate pictures into envelopes and give them their proper names, i.e. the preparation set and the test set. This makes it simpler to bring the pictures into Keras.

1) Setup: In this progression we have to import Keras and different bundles that were going to use in building the CNN. There are different bundles which are incorporated inside it: -

- Sequential is utilized to introduce the neural system.
- Convolution2D is utilized to make the convolutional organize that manages the pictures.
- MaxPooling2D layer is utilized to include the pooling layers.
- Flatten is the capacity that changes over the pooled include guide to a solitary section that is passed to the completely associated layer.
- Dense adds the completely associated layer to the neural system.

2) Initializing the neural system: To instate the neural system we make an object of the Sequential class.

3) Convolution: To include the convolution layer, we call the include work with the classifier article and go in Convolution2D with parameters. The principal contention nbfilter. nbfilter is the quantity of highlight indicators that we need to make. The second and third parameters are measurements of the component indicator grid. It's basic practice to begin with 32 element indicators for CNNs. The following parameter is input shape which is the state of the information picture. The pictures will be changed over into this shape during pre-handling. On the off chance that the picture is highly contrasting it will be changed over into a 2D exhibit and if the picture is hued it will be changed over into a 3D cluster.

4) Pooling: Right now decline the size of the component map. Generally we make a pool size of 2x2 for max pooling. This engages us to diminish the size of the component map while not losing noteworthy picture information.

5) Flattening: In this progression, all the pooled include maps are taken and placed into a solitary vector. The Flatten work straightens all the element maps into a solitary section.

6) Full association: The accompanying stage is to use the vector we got above as the commitment for the neural framework by using the Dense limit in Keras. The principal parameter is output dim which is the quantity of hubs in the shrouded layer. You can decide the most proper number

through experimentation. The higher the quantity of measurements the all the more figuring assets you should fit the model. A typical practice is to pick the quantity of hubs in forces of two. The subsequent parameter is the actuation work. We for the most part utilize the ReLu actuation work in the concealed layer. The following layer we need to include is the yield layer. Right now, utilize the sigmoid enactment work since we anticipate a paired result. In the event that we expected multiple results we would utilize the softmax work. The output_dim here is 1 since we simply anticipate the anticipated probabilities of the classes.

7) Compiling CNN: We at that point accumulate the CNN utilizing the aggregate capacity. This capacity anticipates three parameters: the streamlining agent, the misfortune work, and the measurements of execution. The enhancer is the angle plummet calculation we are going to utilize. We utilize the binary_crossentropy misfortune work since we are doing a double order.

8) Fitting the CNN: We are going to pre-process the pictures utilizing Keras to forestall over fitting. This preparing is known as picture increase. The Keras utility we use for this design is ImageDataGenerator. This capacity works by flipping, rescaling, zooming, and shearing the pictures. The principal contention rescale guarantees the pictures are rescaled to have pixel esteems somewhere in the range of zero and one. horizontal_flip=True implies that the pictures will be flipped on a level plane. Every one of these activities are a piece of the picture expansion.

9) Making a solitary forecast: Now that the model is fitted, we can utilize the foresee technique to make expectations utilizing new pictures. So as to do this we have to pre-process our pictures before we pass them to the anticipate strategy. To accomplish this, we'll utilize a few capacities from numpy. We likewise need to import the picture module from Keras to permit us to stack in the new pictures. The following stage is to stack the picture that we might want to foresee. To achieve this we utilize the load_img work from the picture module. The main contention this capacity takes is the way to the area of the picture and the subsequent contention is the size of the picture. The size of the picture ought to be equivalent to the size utilized during the preparation procedure.

6 RESULT & CONCLUSION

RESULT

The methodology for the current model for self-driving vehicles are just recognize the traffic light signals yet our proposed model, we are distinguishing the speed limitation for the vehicle at an exactness of 92%.

The model successfully detects the traffic light and classifies it according to its type, proposed work serves as a module for navigation system. The use of Faster R-CNN Inception-V2 model via transfer learning improves the accuracy which makes the system reliable for real time application. The outcomes with bounding boxes provide guidelines for real time control actions of the vehicle. The dataset created for the system covers various use cases according to the Indian Signal System. So, the work done in this paper paves the way for realization of self driving cars on Indian streets. In advancement to this, the system can also be optimized for safe driving despite unclear lane markings. Also, it can be equipped with the ability to respond to spoken commands or hand signals from law enforcement or highway safety employees.

The pre-processing of these selective frames is done by resizing them to size of 600 × 800 pixels which are then used for labelling. There are five classes namely red, yellow, straight, left, right which are used for classification of TL. Each frame is labelled manually based on the type of the class it contains. There are several object detection architectures available like Single Shot Multibox Detector (SSD), Faster Region based Convolutional Neural Networks (R-CNN), Region based Fully (R-FCN) which incorporates feature extractors like ResNet-101, Inception-V2, Inception-V3, MobileNet, etc. The selection of architecture and feature extractor is trade-off between speed and accuracy that your application needs.

The combination of traffic signals and road signs determine a visual language that forms set of rules whose interpretation aids for disciplined driving. In the field of autonomous driving the perception of traffic discipline has high industrial potential. The vision system helps to analyze current traffic situation on the road, danger and difficulties near the vehicle, warn and help them for safe, convenient and healthier navigation by providing the useful information. But easier said than done the vision-based object detection and recognition in traffic scenes is still a major challenge to be successfully overcome by the autonomous driving industry. Variation due to lighting and weather condition, random and dynamic scenarios at any moment contributes to the challenges for object detection function.

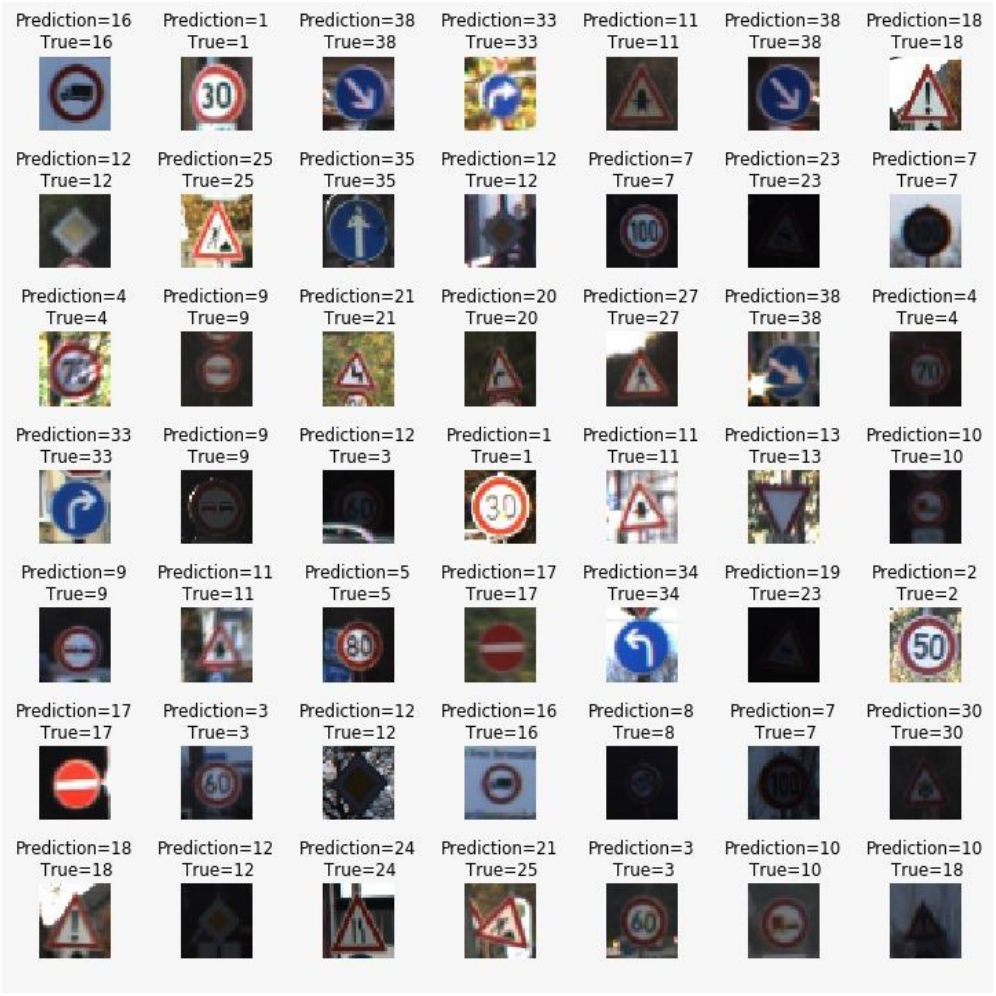

Fig 5.1 Predicted traffic signs

The above figure shows the predicted traffic signs by the car in different scenarios based on the impact faced by our vehicle. The model is able to detect a varied range of traffic signs in different situations.

CONCLUSION

A convnet is a Deep Learning calculation which can take in an info picture, allot significance to it to different viewpoints/protests in the picture and have the option to separate one from the other. The Le net is used in our model to carry out resampling mechanism and to extract all the important parts of the input image. Image augmentation is being carried out to increase the size of our labelled datasets by creating several viewing angles of a single image thereby increasing the number of images. The detected images are being classified based on the traffic signs that they resemble and is used for the movements of self-driving cars. Our model achieves an accuracy of 92% in the training set.

The system incorporates transfer learning based pretrained method wherein Faster R-CNN-Inception-V2 model is used. Transfer learning is a machine learning technique where there is enhancement in learning of a new task by channeling the knowledge through a related task that has formerly been learned. It is a method in deep learning that allows to eliminate the extensive computational and time resources that are essential to develop neural network models by using pre trained models.

The execution of proposed system is distributed into following manner: collecting images of Indian traffic lights, pre-processing images to generate the dataset, training the CNN for detection and recognition of traffic lights and finally validation of model through experimental results.

This algorithm needed to be add some method that can adaptively changing the parameters during day and night adaptively. Because constant parameters can only be used in the same lighting conditions.

The basic idea of this method is the shape and form of local objects can be characterized by distributing the intensity of local gradients or edge directions, even without having the exact precision of the suitability of the gradient or the edge positions. Technically, HOG is implemented by dividing the image into small sections called ‘cell’, then in each cell there is computation of local gradient direction histogram or the edge of the pixel in the cell. The combination of local histograms will form a representation. For better detection of the disturbance of lighting, shadows, and others, contrast normalization can be used in the local response. Contrast normalization is done by accumulating several adjacent local histograms into the larger part of the cell called ‘the block’.

7 REFERENCES

1. Automatic detection of traffic lights changes from red to green and car turn signals in order to improve urban traffic, Julian Balcerek, Adam Konieczka, Tomasz Marciniak, Adam Dąbrowski, Krzysztof Maćkowiak& Karol Piniarski, 2014, Published in: 2014 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA),doi-10.1109/ITSC.1999.821072.
2. Image processing based traffic sign detection and recognition with fuzzy integral, CananTaştimur, Mehmet Karaköse, Yavuz Çelik& Erhan Akın, 2016, Published in: 2016 International Conference on Systems, Signals and Image Processing (IWSSIP), doi-10.1109/IWSSIP.2016.7502715.
3. An efficient framework for recognizing traffic lights in night traffic images, Bo Fan, Weiyao Lin &Xiaokang Yang, 2012, Published in: 2012 5th International Congress on Image and Signal Processing, doi-10.1109/CISP.2012.6469638.
4. A gradient-domain image enhancement method for traffic signs in nighttime surveillance, Simin Wang, Xiaoguang Li, Hui Zhang & Li Zhuo, 2017, Published in: 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI) doi-10.1109/CISP-BMEI.2017.8301943
5. Real-time image processing approach to measure traffic queue parameters, M. Fathy & M.Y. Siyal, 1995, Published in: IEE Proceedings - Vision, Image and Signal Processing (Volume: 142, Issue: 5, Oct 1995), doi-10.1049/ip-vis:19952064
6. A measurement method of pedestrian traffic flows by use of image processing and its application to a pedestrian traffic signal control, Y. Iwasaki, 1999, Published in: Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383), doi-10.1109/ITSC.1999.821072.
7. Detection and recognition of traffic signs based on RGB to red conversion, Mohit BhairavMahatme& Mrs. Sonia Kuwelkar, 2017, Published in: 2017 International Conference on Computing Methodologies and Communication (ICCMC), doi-10.1109/ICCMC.2017.8282728.
8. Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network, Hee Seok Lee & Kang Kim, 2018, Published in: IEEE Transactions on Intelligent Transportation Systems (Volume: 19, Issue: 5, May 2018), doi-10.1109/TITS.2018.2801560.
9. Traffic sign detection and classification using colour feature and neural network, Md. Abdul Alim Sheik, Alok Kole & Tanmoy Maity, 2016, Published in: 2016 International

Conference on Intelligent Control Power and Instrumentation (ICICPI), doi-10.1109/ICICPI.2016.7859723

10. A traffic sign detection algorithm based on deep convolutional neural network, XiongChangzhen, Wang Cong, Ma Weixin& Shan Yanmei, 2016, Published in: 2016 IEEE International Conference on Signal and Image Processing (ICSIP), doi-10.1109/SIPROCESS.2016.788

8 APPENDICES

A) Source Code

```
import pickle

import seaborn as sns

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import random

with open("./traffic-signs-data/train.p", mode='rb') as training_data:

    train = pickle.load(training_data)

with open("./traffic-signs-data/valid.p", mode='rb') as validation_data:

   valid = pickle.load(validation_data)

with open("./traffic-signs-data/test.p", mode='rb') as testing_data:

   test = pickle.load(testing_data)

X_train, y_train = train['features'], train['labels']

X_validation, y_validation = valid['features'], valid['labels']

X_test, y_test = test['features'], test['labels']

X_train.shapey_train.shape

i = 1001
```

```
plt.imshow(X_train[i]) # Show images are not shuffled

y_train[i]

from sklearn.utils import shuffle

X_train, y_train = shuffle(X_train, y_train)

X_train_gray = np.sum(X_train/3, axis=3, keepdims=True)

X_test_gray  = np.sum(X_test/3, axis=3, keepdims=True)

X_validation_gray  = np.sum(X_validation/3, axis=3, keepdims=True)

X_train_gray_norm = (X_train_gray - 128)/128

X_test_gray_norm = (X_test_gray - 128)/128

X_validation_gray_norm = (X_validation_gray - 128)/128

i = 610

plt.imshow(X_train_gray[i].squeeze(), cmap='gray')

plt.figure()

plt.imshow(X_train[i])

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D, Dense, Flatten, Dropout

from keras.optimizers import Adam

from keras.callbacks import TensorBoard

from sklearn.model_selection import train_test_split

image_shape = X_train_gray[i].shape

cnn_model = Sequential()

cnn_model.add(Conv2D(filters=6,kernel_size=(5,5),activation='relu',input_shape=(32,32,1))
)
```

```
    cnn_model.add(AveragePooling2D())
    cnn_model.add(Conv2D(filters=16, kernel_size=(5, 5), activation='relu'))
    cnn_model.add(AveragePooling2D())
    cnn_model.add(Flatten())
    cnn_model.add(Dense(units=120, activation='relu'))
    cnn_model.add(Dense(units=84, activation='relu'))
    cnn_model.add(Dense(units=43, activation = 'softmax'))
    cnn_model.compile(loss='sparse_categorical_crossentropy',optimizer=Adam(lr=0.001),metri
    cs =['accuracy'])
    history = cnn_model.fit(X_train_gray_norm,
    y_train,
batch_size=500,
nb_epoch=50,
                        verbose=1,
    validation_data = (X_validation_gray_norm,y_validation))
    accuracy = history.history['acc']
    val_accuracy = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(len(accuracy))
    plt.plot(epochs, accuracy, 'bo', label='Training Accuracy')
    plt.plot(epochs, val_accuracy, 'b', label='Validation Accuracy')
    plt.title('Training and Validation accuracy')
    plt.legend()
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_true, predicted_classes)

plt.figure(figsize = (25,25))

sns.heatmap(cm, annot=True)

L = 7

W = 7

fig, axes = plt.subplots(L, W, figsize = (12,12))

axes = axes.ravel() #

for i in np.arange(0, L * W):

    axes[i].imshow(X_test[i])

    axes[i].set_title("Prediction={}\n True={}".format(predicted_classes[i], y_true[i]))

  axes[i].axis('off')

plt.subplots_adjust(wspace=1)
```

B) Screenshots

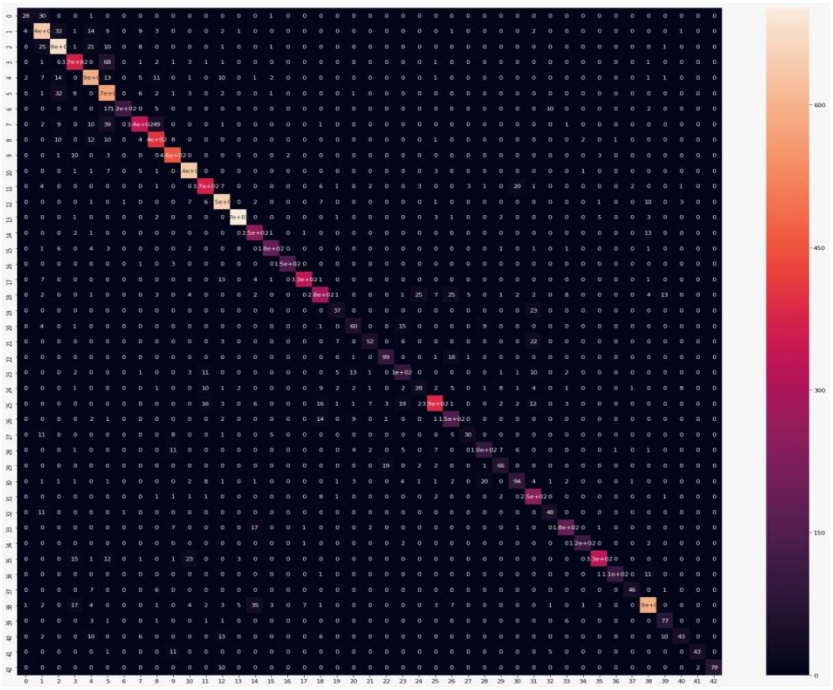

Fig B.1 Heat Map

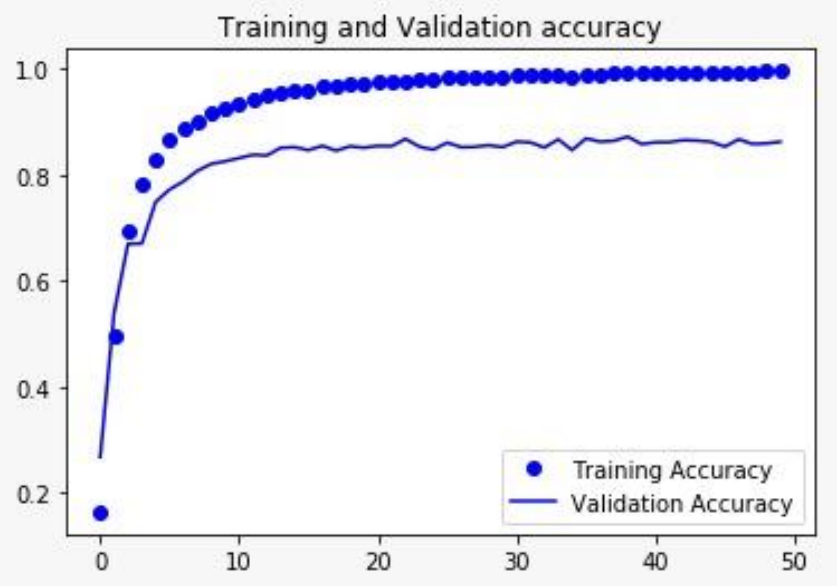

Fig B.2 Training and validation accuracy

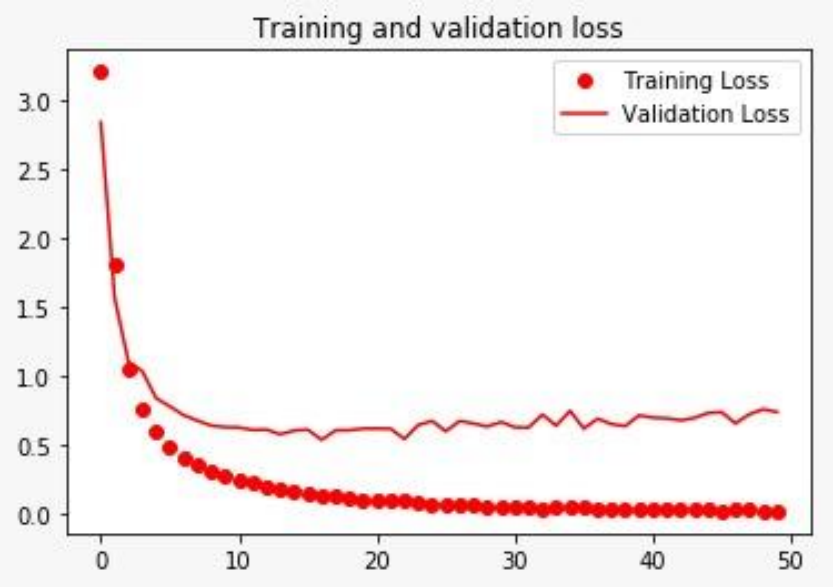

Fig B.3 Training and validation loss