

PROFESSIONAL TRAINING REPORT

at

**Sathyabama Institute of Science and
Technology**

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the
award of Bachelor of Engineering Degree in
Computer Science and Engineering By

TANNIRU DURGA PAVAN

(38110143)

MANEPALLI VAMSI VENKAT

(38110613)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

MARCH 2022



SATHYABAMA
INSTITUTE OF SCIENCE AND
TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI– 600119
www.sathyabama.ac.in

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Vamsi Venkat Manepalli (Reg. No. 38110613)**, **Tanniru Durga Pavan (Reg. No. 38110143)** who carried out the project entitled “**Traffic Sign Prediction using Machine Learning for Autonomous Drivers**” under my supervision from **November 2021 to March 2022**

Internal Guide

Dr.L. Mary Gladence, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Dr. L. LAKSHMANAN, M.E., Ph.D.,

Submitted for Viva voce Examination held on

Internal Examiner

External

**Examiner
DECLARATION**

I, **Manepalli Vamsi Venkat (Reg. No. 38110613), Tanniru Durga Pavan (38110143)** hereby declare that the project report entitled **“TRAFFIC SIGN PREDICTION USING MACHINE LEARNING FOR AUTONOMOUS DRIVERS”** done by me under the guidance of **Dr. L . Mary Gladence, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

*M.vamsi venkat
t.durga pavan*

PLACE : CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D, Dean, School of Computing** and

Dr. S. Vigneshwari, M.E., Ph.D. and **Dr. L. Lakshmanan, M.E., Ph.D , Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide of **Dr. L. Mary Gladence , M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Traffic Sign Prediction is crucial in the development of Intelligent Vehicles. To foster savvy vehicles for street security, it is important to know and know the indications of the street. The calculation is the old procurement of LeNet-5 CNN. The method was presented utilizing Gabor's presentation, trailed by the standard revolution later the reconciliation of parts. The best strategy to use here is the Adams technique. Shade and Saturation The worth of shading mirrors the attributes of speed discovery and low light. The street recognizable proof procedure is being tried with the assistance of German Vehicle Identification. This project has a UI framework designed using HTML, CSS and both the model, framework are deployed using Flask. The UI framework takes required inputs to predict the water quality and passes the inputs to machine learning model and predicted output is displayed on the page

Firstly, Traffic Signs are effectively detected based on the shape features.

Secondly, the model is considerably improved on the basis of the classic Sequential Model.

Finally, the Traffic Sign Prediction and recognition experiments are conducted based on the German Traffic Sign Recognition Benchmark. Experimental results show that the accurate recognition rate of traffic signs reaches 97.49%.

	TABLE OF CONTENTS	
Chapter No.	TITLE	Page No.
	ACKNOWLEDGEMENT	
	ABSTRACT	
1	INTRODCUTION	1
1.1	What is Machine Learning	1

1.2	What is deep learning	2
1.3	Advantages of machine learning	3
1.4	Tasks taken up and problem definition	4
1.4.1	Problem definition	4
2	Literature Survey	5
3	Methodology And Learning	6
3.1	Convolutional layer	6
3.2	Pooling layer	6
3.3	Image Segmentation Module	7
3.4	Traffic Sign Detection Module	7
3.5	Traffic Sign-recognition	7
3.6	Cnn architecture	8
3.4	Traffic sign recognition	8
3.4.1	LeNet-5 CNN Model	8
3.4.2	Extended LeNet-5 Model	9
4	Work Done	10
4.1	Gathering and exploring data	10
4.2	Importing pre-requisite	11
4.3	Building a cnn model	13
4.3.1	Architecture of the model	13
4.4	Training and testing the dataset	14
4.5	Plotting accuracy	15
4.6	Testing the model with dataset	16
4.7	Deploying the model	16
4.7.1	Flask	16
4.7.2	Tool used	17
4.7.3	Installation	17
5	Results and Discussions	18
6	References	21

7	Source Code	22
---	-------------	----

1.Introduction:

1.1. What is Machine Learning?

Definition-

A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**. ~ Tom M. Mitchell.

Machine learning (ML) is rapidly changing the world, from diverse types of applications and research pursued in industry and academia. Machine learning is affecting every part of our daily lives. From voice assistants using NLP and machine learning to make appointments, check our calendar and play music, to programmatic advertisements — that are so accurate that they can predict what we will need before we even think of it.

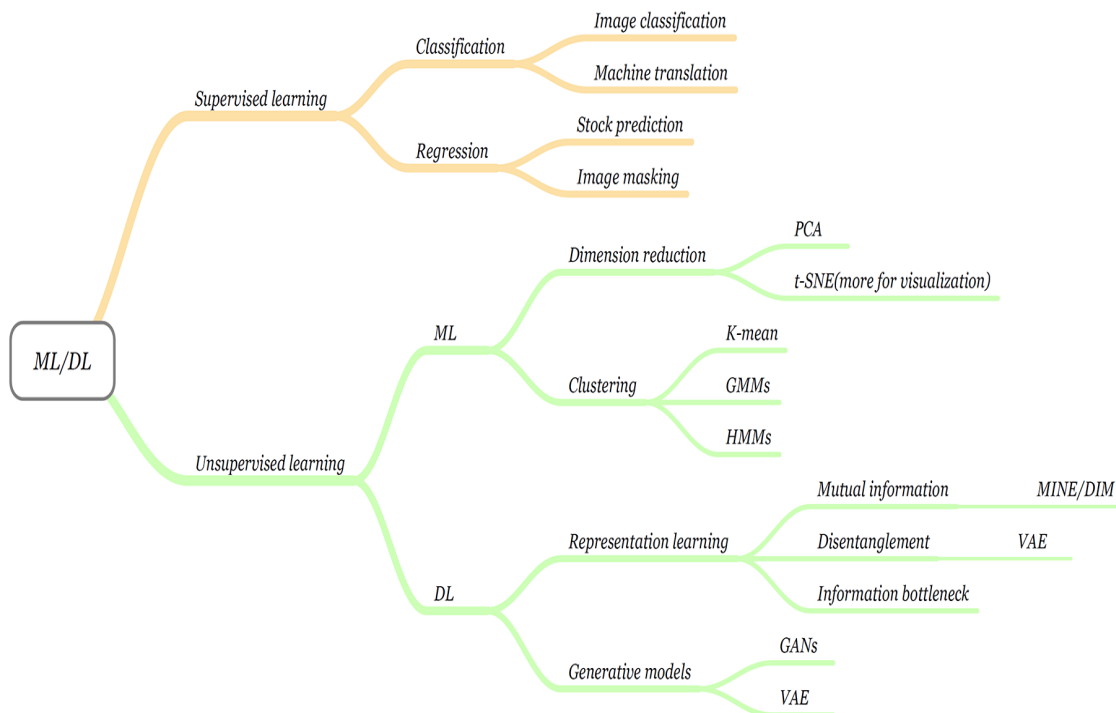


Fig-1-Structure of ML/DL

1.2. What is Deep Learning?

Deep learning is a subset of machine learning in which layered neural networks, combined with high computing power and large datasets, can create powerful machine learning models.

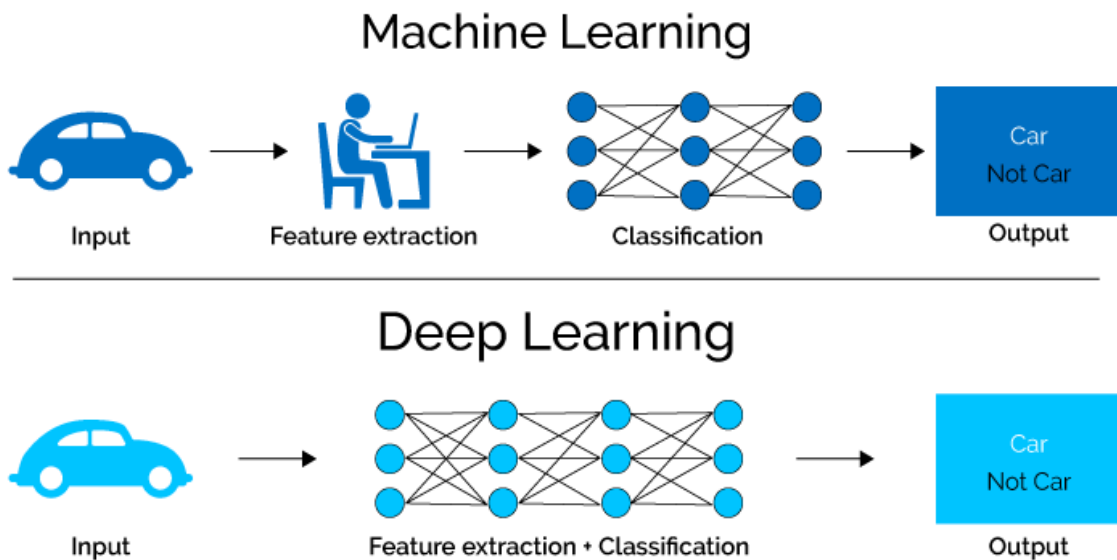


Fig-2

Why do we prefer Python to implement Machine Learning Algorithms?

Python is a popular and general-purpose programming language. We can write machine learning algorithms using Python, and it works well. The reason why Python is so popular among data scientists is that Python has a diverse variety of modules and libraries already implemented that make our life more comfortable.

Some exciting Python libraries which make implementation easy:

1. **Numpy:** It is a math library to work with n-dimensional arrays in Python. It enables us to do computations effectively and efficiently.

2. Matplotlib: It is a trendy plotting package that provides 2D plotting as well as 3D plotting.

3. Scikit-learn: It has most of classification, regression and clustering algorithms.

4. OpenCv- It is a python library in which it manipulates the image process. it is converted from numpy arrays and it can easily integrate with numpy such as scipy and matplotlib

1.3. Advantages of Machine Learning:

1. Easily identifies Trends and Patterns:-

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation):-

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement:-

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.



Fig-3-advantages of ml

1.4. TASKS TAKEN UP AND PROBLEM DEFINITION

- Learnt the fundamentals of Python, Machine Learning and basics of Deep Learning.
- Basics of Python Programming Language:-
 1. NumPy
 2. Pandas
 3. Basic python syntaxes
- Fundamentals of Machine Learning:-
 1. Machine Learning terminology and overview
 2. Regression Problems
 3. Classification Problems
- Basics of Deep Learning:-
 1. Basics of Neural Networks with some examples

1.4.1. Problem Definition:-

Developing an efficient Machine Learning Model which would predict various Traffic Signs.

This process is done in four steps:-

1. Gathering and exploring the dataset
2. Building a CNN Model
3. Training and testing the model
4. Deploying the model

2. Literature Survey

[1]Traffic lights and groupings assume a vital part in driving. As of late, different methodologies have been proposed to resolve this issue, however the exhibition of this calculation actually should be improved to meet continuous functional necessities. The new proof in this paper presents a proof and order strategy in view of the Convolution Neural Network and Machine Vector Machine (CNN-SVM). Thusly, CNC acquainted the YCbCr shading space with diminish the shading channel to eliminate objects.[2] The SVM classifier is utilized to arrange things in view of erased things. The review depended on constant information and video got from an ordinary machine. The consequences of the review show that our strategy is on normal 98.6% better than the advanced technique by deciding and ordering the light sign boundaries.

Arranging is finished utilizing the Extreme Learning Machine (ELM) calculation. Framework execution was evaluated on both the German Motor Vehicle and the Belgian Vehicle guidelines.[3]The consequences of the review show that everything gives high honesty, and the blend of the three elements is well correlative, rapidly unmistakable, and reasonable for appropriate use .[4]Our technique incorporates street signs and lines. In the cognizance module, the red and blue tones are improved and MSER is performed to further develop the competitor's street sign region. Bayesian and DtB models are utilized to comprehend street signs.[5]Street markings were set up and carried out through fixed pipelines. By and large, this action is assessed by the outcomes acquired to demonstrate the reality of the opposition without preparing data. [6] Until this point in time, most picture acknowledgment advances utilize drawbacks as well as the intricacy of pictures. So different analysts are attempting to work on the calculation and improve it and better.[7] At first, the conventional standards of the convulsive neural organization are momentarily presented. Many picture handling programs are being delivered. At long last, our new work to address the intricacies, scientific difficulties, and afterward the ability issues confronting the Convolution Neural Network.

[8]We utilize this information to all the more likely distinguish other nation/area street signs through top to bottom review, utilizing inside and out relocation procedures to execute the information acquired from cutting edge exercises and an enormous number of street signs in a given nation/locale.[9] This permits clients to utilize informational indexes previously made from different districts to assist with characterizing the ideal objective, liberating clients from the weight of information assortment and distinguishing proof.[10] We propose three methods for moving preparation, two of which are more exact than top to bottom learning. This study exhibits that the trading of information between inside and out investigations can give more trustworthiness in distinguishing street signs than top to bottom street sign acknowledgment models.[11] The reason for this article is to recognize traffic lights progressively, ie to recognize the sorts of street signs that show up in the picture entered during quick handling. To accomplish this, we want the quickest identification module, which is multiple times quicker than the current recognition module.[12] Our understanding module depends on the capacity to eliminate plaque and group structures so as to show tone and shading HOG.

This technique centres around perceiving and remembering one-on-one contact pictures that are not utilized in worldwide tasks.[13]Dissimilar to past examinations, we have fostered a coordinated computation strategy to distinguish, screen, and recognize street signs with a mono camera introduced in a moving vehicle. The principle commitments of this record are triple: 1) another strategy for apportioning the region before the street sign has been utilized in this report to further develop understanding execution utilizing applicable data;) and 3) direct based measures were gotten to get a steady result arrangement.

3. METHODOLOGY AND LEARNING

Traffic Sign Prediction is crucial in the development of Intelligent Vehicles.

An improved Traffic Sign Prediction Algorithm for Intelligent Vehicles is proposed to address problems such as how easily affected Traditional Traffic Sign predicted is by the environment, and poor real-time performance of deep learning-based methodologies for traffic sign recognition.

Firstly, Traffic Signs are effectively detected based on the shape features.

Secondly, the model is considerably improved on the basis of the classic Sequential Model by using Gabor kernel as the initial Convolutional Kernel, adding the batch normalization processing after the Pooling Layer and selecting Adam Method as the Optimizer Algorithm.

3.1. COVOLUTIONAL LAYERS :

This is the first layer and one of the main building blocks of a Convolutional Neural Networks (CNNs).

They hold the raw pixel values of the training image as input i.e. extract features from it.

This layer ensures the spatial relationship between pixels by learning image features using small squares of input data.

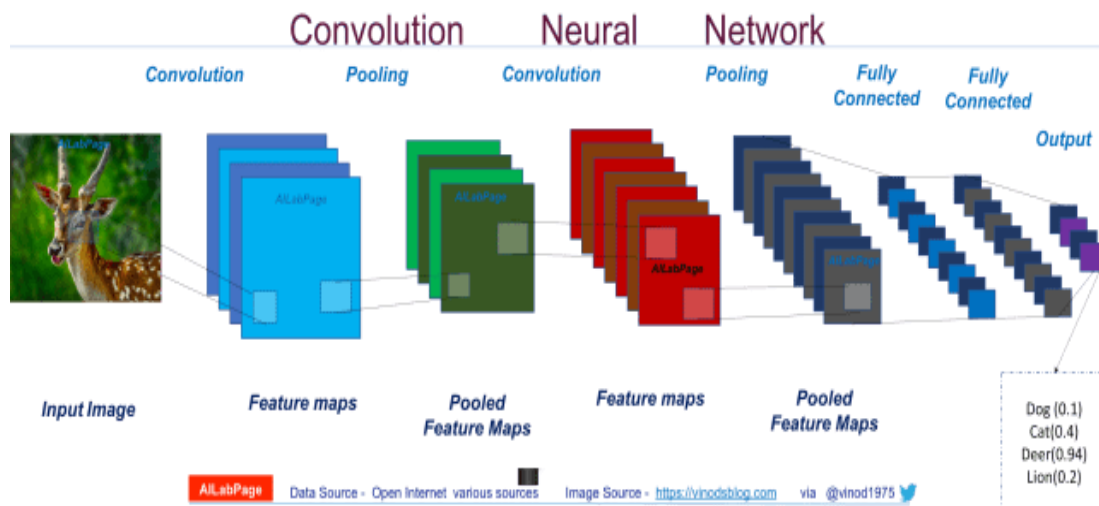


Fig-4-layer transformation

3.2. POOLING LAYER :

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

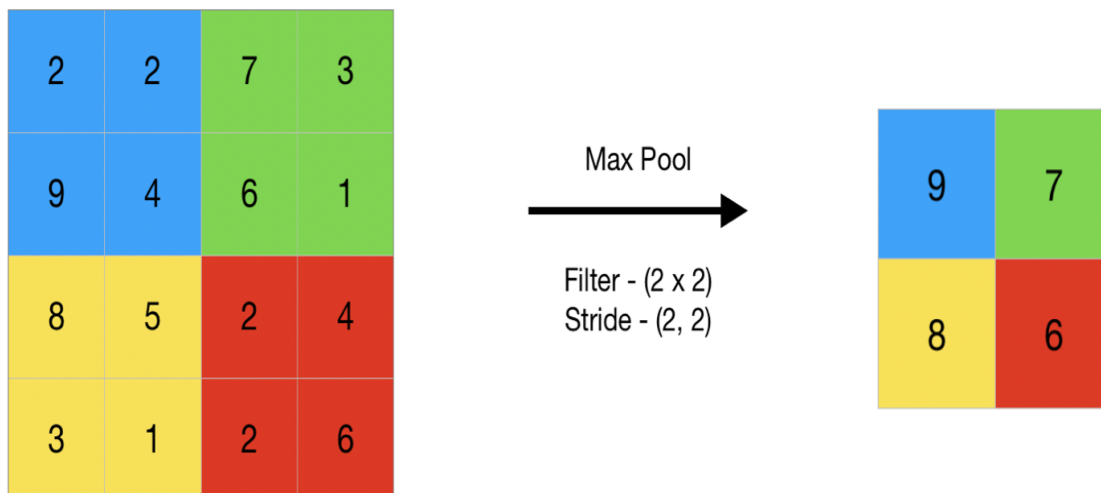


Fig-5-pooling layer

Image Pre-Processing Module

Utilizing pictures is a typical method for changing over pictures prior to involving them in drawing. This incorporates changes, showings, and added shadows, yet not simply restricted ones. Now and again, it very well might be smart to eliminate the data from the picture to save time or decrease the danger. Change the direction to the picture.

3.3. Image Segmentation Module:

Sharing is a picture sharing procedure accessible in an assortment of pixels called obstruction, which decreases the size of the picture and picture learning. Disallowance is a method that utilizes marginal strategies to change over a picture into a sorry excuse for a picture in two sections.

3.4. Traffic Sign Detection Module:

Street signs and signs are a significant piece of the expert level, like driving and driving. This straightforwardly helps the driver or framework to precisely distinguish and recognize traffic signs. In this module, we address the issue of recognizing and presentation of street signs. Since there are such countless little yet profound things, we requested methodologies to comprehend and separate utilizing the RCNN (Region convolutional neural organization) veil, in view of exploration. This establishment gives a first-rate association that looks for countless classes to view as simple and separating images.

3.5. Traffic Sign-recognition:

Improvement Learning is a sub domain of Machine Learning that incorporates Convolutional Neural Networks. Street Signs (TSRs) are street finishes paperwork for vehicles. "speed" or "youngsters" or "front". This is the genuine piece of ADAS. It utilizes outlines to distinguish traffic signs.

The camera was displayed to the staggering vehicles used to give road signs. The computation depends on the picture of the road sign showed on the camera. In one spot, road signs don't look like shade, shape, or shading. Prior to managing obscure road signs that have been amended and recognized, they have been affirmed by different street signs.

3.6. CNN Architecture:

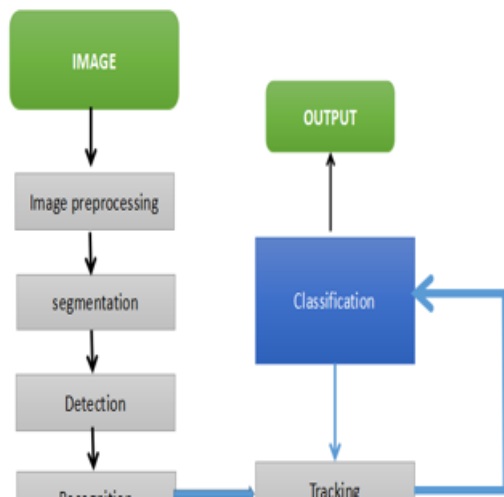


Fig-6-architecture

A Convolutional Neural Network (Conv Net/CNN) is a Deep Learning set of rules that can soak up an entire photograph, assign importance (learnable weights and biases) to diverse aspects/items within the photograph and have the ability to distinguish one from the other. The pre-processing required in a Conv Net is a good deal less compared to different category algorithms. While in primitive techniques filters are hand-engineered, with sufficient training, ConvNets have the potential to study those filters/characteristics.

CNN is especially utilized in picture evaluation applications like Image recognition, Object detection & Segmentation

CNNs are used for picture classification and popularity due to its remarkable accuracy. It changed into proposed via way of means of a scientist Yann LeCun with inside the past few 90s, whilst he changed into stimulated from the human visible notion of spotting things

3.7. TRAFFIC SIGN RECOGNITION

3.7.1. LeNet-5 CNN Model

The most ideal way to do this is to isolate the traffic signs utilizing a bookkeeping page to rapidly recognize the traffic signs. Similarly, Conv. NN can't kill fingerprinting work. Crafted by the human brain is firmly connected with the manner in which he imparts, thinks, and acts, and along these lines follows up on the force of the street that meets the guidelines of the street. This segment breaks down old LeNet-5 issues and adds Conv.NN advantages to program goal. Specifically, the LeNet-5 enjoys the benefit of acknowledgment and translation. Regardless, it is hard to know the specific degree of groundwork for the afternoon, the arrangement of the channel.

3.7.2. Extended LeNet-5 Model

Recognizing street signs relies upon the size of the middle and the utilization of countless estimations, just as a restricted comprehension of strange and troublesome vehicles. Conv.NN shows the data in

the pictures and decides the outcomes through the classifications made in view of the picture line. This correlation demonstrates that Conv.NN has a genuine capacity to be perceived. The human brain is encircled by astounding things that associate, recognize and perceive the

indications of the street. Here, we audit the LeNet-5 glitches and work on the Conv.NN exceptional highlights map. Organizations and ROI and essential data. Thusly, you should initially manage the pictures.

4.Work Done

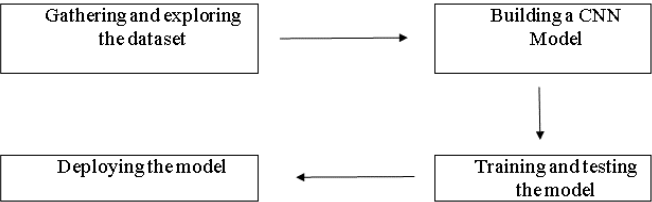


Fig-7-implementation

4.7. GATHERING AND EXPLORING DATA:-

The dataset contains more than 50,000 images of different traffic signs. It is further classified into 43 different classes. The dataset is quite varying; some of the classes have many images while some classes have few images. The size of the dataset is around 600mb. The dataset has a train folder which contains images inside each class and a test folder which we will use for testing our model.



Fig-8-bench maark of dataset

Class 0-210	Class 1-2220	Class 2-2250	Class 3-1410	Class 4-1980	Class 5-1860	Class 6-420	Class 7-1440
Class 8-1410	Class 9-1470	Class 10-2100	Class 11-1320	Class 12-2100	Class 13-2160	Class 14-780	Class 15-630
Class 16-420	Class 17-1110	Class 18-1200	Class 19-210	Class 20-360	Class 21-330	Class 22-390	Class 23-510
Class 24-270	Class 25-1500	Class 26-600	Class 27-240	Class 28-540	Class 29-270	Class 30-450	Class 31-780
Class 32-240	Class 33-689	Class 34-420	Class 35-1200	Class 36-390	Class 37-210	Class 38-2070	Class 39-300
Class 40-360	Class 41-240	Class 42-240					

4.8.IMPORTING PRE-REQUISITE

The screenshot shows a code editor with a dark theme. The top menu bar includes File, Edit, View, Navigate, Code, and Refactor. The breadcrumb navigation shows Traffic > Web_App > requirements.txt. The editor has two tabs: Traffic_App.py and requirements.txt. The requirements.txt file contains the following content:

```
1 numpy
2 Pillow
3 scikit-learn==0.23.2
4 keras==2.3.1
5 tensorflow==1.14.0
6 flask
7 Werkzeug
8 gunicorn
```

11

```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: import matplotlib.pyplot as plt

In [4]: import cv2

In [5]: import tensorflow as tf

In [6]: from PIL import Image

In [7]: import os

In [8]: os.chdir('D:/Traffic')

In [9]: from sklearn.model_selection import train_test_split

In [10]: from keras.utils import to_categorical

Using TensorFlow backend.
```

Fig-11-importing the pre requisites

1. NumPy – NumPy is a python library used for mathematical computations.
2. Pandas – Pandas is data manipulation package tool whose main data structure is Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables.
3. Matplot- Matplot Library is used for Data Visualization.
4. OpenCv- Library of Python bindings designed to solve computer vision problems. All the OpenCV array structures are converted to and from Numpy arrays.
5. Tensor flow- It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of Tensor Flow.
6. PIL - The PIL library is used to open image content into an array.

7. Scikit Learn- Used for splitting the data into train data and test data.
8. Keras- Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models.
9. Os module – For manipulating the paths in local module.

4.9. BUILDING A CNN MODEL:-

To classify the images into their respective categories, we will build a CNN model (**Convolutional Neural Network**). CNN is best for image classification purposes.

4.9.2. ARCHITECTURE OF THE MODEL:-

2 Conv2D layer (filter=32, kernel size=(5,5), activation="relu")
MaxPool2D layer (pool size=(2,2))
Dropout layer (rate=0.25)
2 Conv2D layer (filter=64, kernel size=(3,3), activation="relu")
MaxPool2D layer (pool size=(2,2))
Dropout layer (rate=0.25)
Flatten layer to squeeze the layers into 1 dimension
Dense Fully connected layer (256 nodes, activation="relu")
Dropout layer (rate=0.5)
Dense layer (43 nodes, activation="softmax")

```

#Convolutional Model!
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
# We have 43 classes that's why we have defined 43 in the dense
model.add(Dense(43, activation='softmax'))

```

Fig-12-architecture of system

4.10. TRAINING AND TESTING THE DATASET:-

- Our dataset is acquired from kaggle org. and the entire dataset is splitted into two classification termed as “Train_set” ‘n’ “Test_set”. The entire dataset is of 40+ folders in which there 40,000+ images. The way they are organized in the folders is In each folders there exist a same sign in different perspective like one in top view, front view. One in a rainy day, foggy day etc,.
- This make my algorithm more accurate towards detecting the traffic sign with less flaws.
- After constructing the version architecture, we then teach the version the use of version.fit (). We attempted with batch length 32 and sixty four. Our version achieved higher with sixty four batch length. And after 15 epochs the accuracy became stable.
- Our dataset includes a check folder and in a check.csv file, we've got the info associated with the photo direction and their respective magnificence labels. We extract the photo direction and labels the use of pandas. Then to expect the version, we ought to resize our pix to 30×30 pixels and make a numpy array containing all photo data. From the sklearn metrics, we imported the accuracy rating and discovered how our version expected the real labels.

```
In [25]: #Training for 20 epochs!
epochs = 20
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))

Train on 62734 samples, validate on 15684 samples
Epoch 1/20
62734/62734 [=====] - 277s 4ms/step - loss: 1.2417 - accuracy: 0.6740 - val_loss: 0.1857 - val_acc
uracy: 0.9525
Epoch 2/20
62734/62734 [=====] - 272s 4ms/step - loss: 0.3744 - accuracy: 0.8908 - val_loss: 0.1330 - val_acc
uracy: 0.9631
Epoch 3/20
62734/62734 [=====] - 268s 4ms/step - loss: 0.2912 - accuracy: 0.9177 - val_loss: 0.0511 - val_acc
uracy: 0.9866
Epoch 4/20
62734/62734 [=====] - 270s 4ms/step - loss: 0.2559 - accuracy: 0.9311 - val_loss: 0.0619 - val_acc
uracy: 0.9839
Epoch 5/20
62734/62734 [=====] - 266s 4ms/step - loss: 0.2383 - accuracy: 0.9358 - val_loss: 0.0556 - val_acc
uracy: 0.9864
Epoch 6/20
62734/62734 [=====] - 266s 4ms/step - loss: 0.2181 - accuracy: 0.9422 - val_loss: 0.0477 - val_acc
uracy: 0.9872
Epoch 7/20
62734/62734 [=====] - 267s 4ms/step - loss: 0.2205 - accuracy: 0.9422 - val_loss: 0.0364 - val_acc
uracy: 0.9890
Epoch 8/20
62734/62734 [=====] - 269s 4ms/step - loss: 0.2435 - accuracy: 0.9398 - val_loss: 0.0374 - val_acc
uracy: 0.9891
Epoch 9/20
62734/62734 [=====] - 267s 4ms/step - loss: 0.2176 - accuracy: 0.9463 - val_loss: 0.0357 - val_acc
uracy: 0.9895
Epoch 10/20
62734/62734 [=====] - 267s 4ms/step - loss: 0.2114 - accuracy: 0.9484 - val_loss: 0.0284 - val_acc
uracy: 0.9927
```

Fig-13-training of dataset

4.11. PLOTTING ACCURACY:-

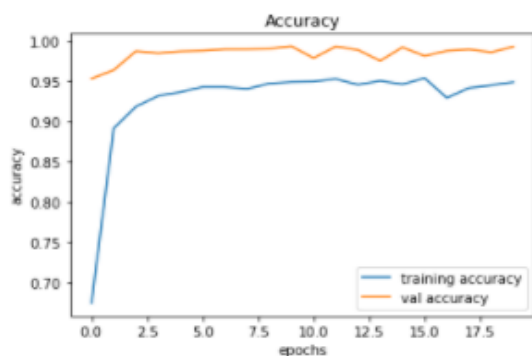


Fig-14-accuracy graph

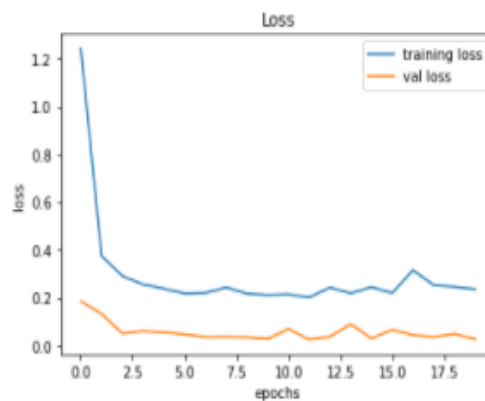


Fig-14.1-loss of dataset

In this graph it's shows before the testing and tranning of a dataset of epochs of 15 which gives the best results

The general exactness is 96.72%. This was around 98.84% of the populace. Nonetheless, with such definitions, we can adjust the plan to get more natty gritty data on the best way to utilize this capacity. With the assistance of a recording gadget, the casing can advise the driver when calling.

4.12. TESTING THE MODEL WITH DATASET:-

Our dataset contains a test folder and in a test.csv file, we have the details related to the image path and their respective class labels. We extract the image path and labels using pandas. Then to predict the model, we have to resize our images to 30×30 pixels and make a numpy array containing all image data. From the sklearn metrics, we imported the accuracy score and observed how our model predicted the actual labels.

Now TESTING on TEST DATA!

```
In [29]: def testing(testcsv):
        y_test = pd.read_csv(testcsv)
        label = y_test["ClassId"].values
        imgs = y_test["Path"].values
        data=[]
        for img in imgs:
            image = Image.open(img)
            image = image.resize((30,30))
            data.append(np.array(image))
        X_test=np.array(data)
        return X_test,label

In [30]: X_test, label = testing('Test.csv')

In [31]: Y_pred = model.predict_classes(X_test)
        Y_pred

Out[31]: array([16,  1, 38, ...,  2,  7, 10], dtype=int64)
```

ACCURACY with TEST DATA!

```
In [32]: from sklearn.metrics import accuracy_score
        print(accuracy_score(label, Y_pred))

0.9612034837688044
```

Fig-15-testing of dataset

- We have achieved an accuracy of almost 96%.

4.13. DEPLOYING THE MODEL:-

We have seen the training and testing part till now, here on we will convert this deep learning model into real world application we use **flask**

4.13.2. FLASK:-

Flask is an internet framework. This method flask gives you with tools, libraries and technology that let you construct an internet utility. This net utility may be a few net pages, a blog, a wiki or cross as huge as an internet-primarily based totally calendar utility or a industrial website.

Flask is a light-weight WSGI net utility framework. It is designed to make getting began out brief and easy, with the capacity to scale as much as complicated applications.

4.13.3. TOOL USED:- {IDE}

PyCharm:

PyCharm is an integrated development environment(IDE) used in computer programming, specifically for the Python language. It is developed by the Jet Brains. PyCharm is cross-platform, with Windows, macOS and Linux versions.

4.13.4. INSTALLATION:

Within the activated environment, use the following command to install Flask:

```
$ pip install Flask
Flask is now installed.
```



Fig-16-upload button

By clicking the upload button it allows us to upload a picture.

UPLOAD BUTTON FUNCTION

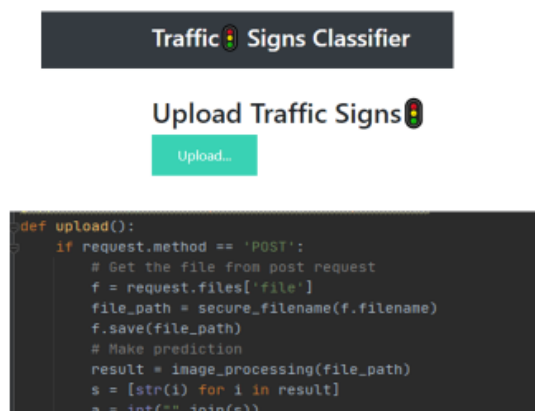


Fig-17-code for upload button

5. Results and Conclusion

The news program incorporates more than 50,000 pictures of different bulletins. It is isolated into 43 distinct classes. The informational collection is altogether different; Some classes have more pictures, while others have less pictures. The information size is 600mb. The informational index

3. contains a train station containing pictures of every classification and a test stop that we can use to gauge the example.

This data set name is know as german traffic sign detection which is taken from kaggle



Here's the final output of my project
**'TRAFFIC SIGN RECOGNITION
USING DEEP LEARNING FOR
AUTONOMOUS DRIVERLESS
VEHICLES'** shown in below figures

Fig-18

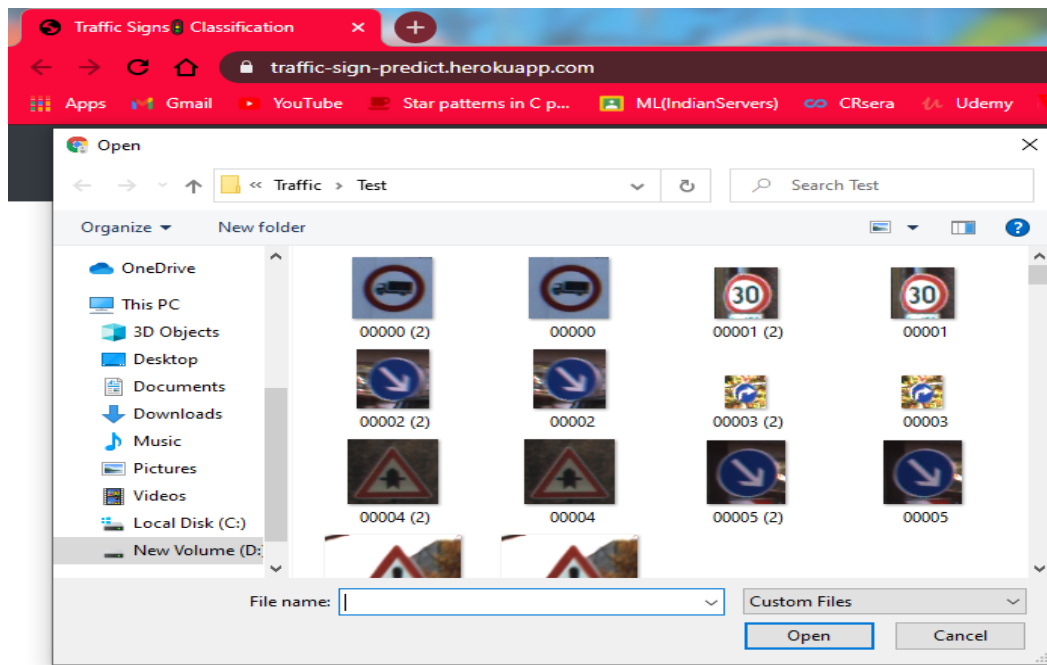
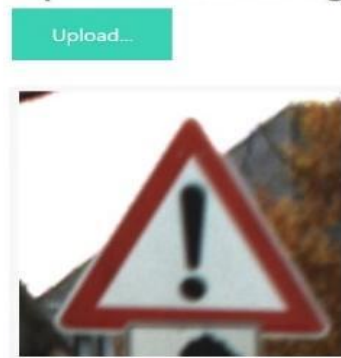


Fig-18.1/ Fig-18.2



Upload Traffic Signs



Predicted Traffic Sign is: General caution

Get Traffic Signs

Fig-18.3

Unsupervised Learning would generally give better performance and results for large data sets. If you have a huge data set easily available, go for deep learning techniques. You also have learned Reinforcement Learning and Deep Reinforcement Learning. You now know what Neural Networks are, their applications and limitations.

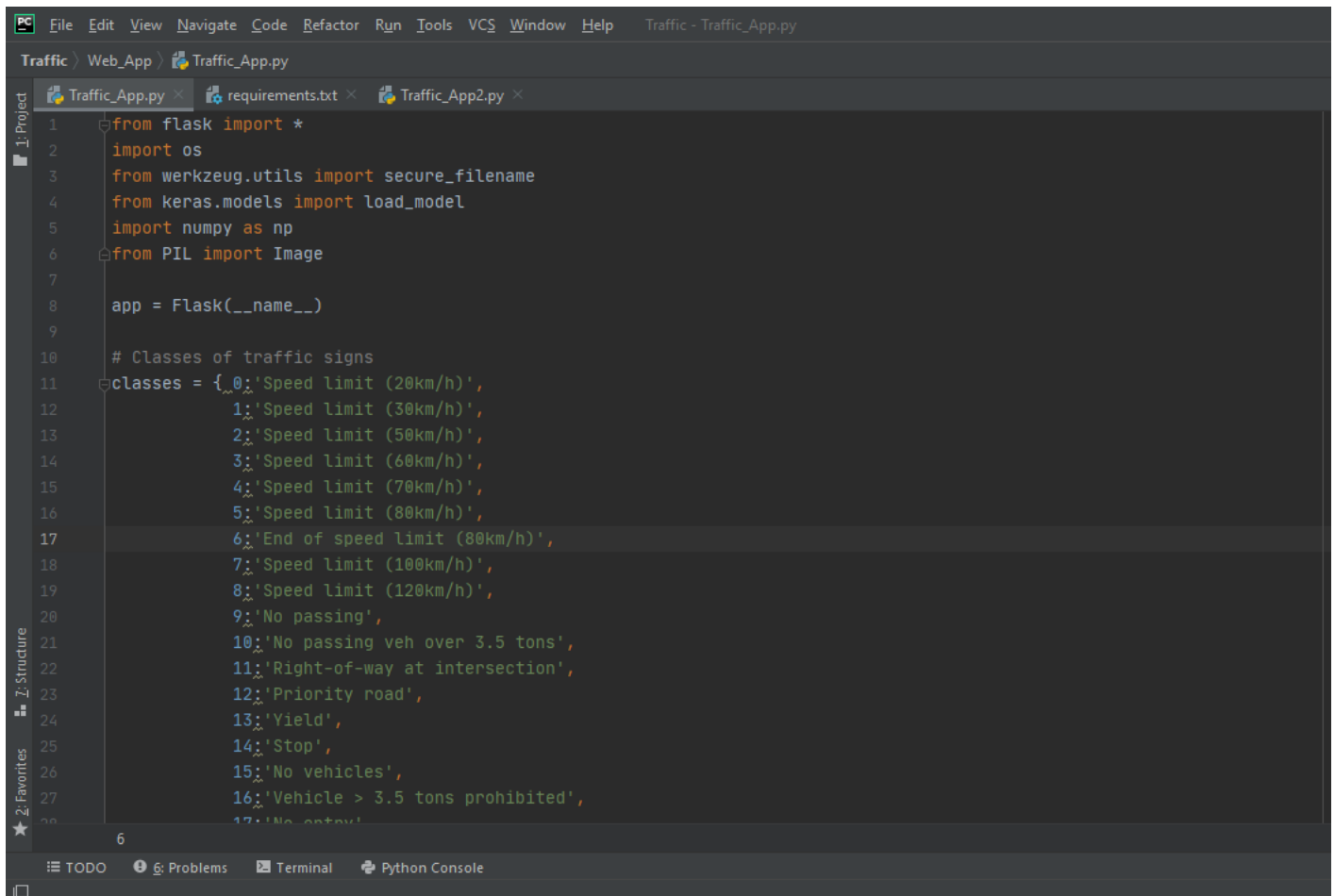
Finally, when it comes to the development of machine learning models of your own, you looked at the choices of various development languages, IDEs and Platforms. Next thing that you need to do is start learning and practicing each machine learning technique. The subject is vast, it means that there is width, but if you consider the depth, each topic can be learned in a few hours. Each topic is independent of each other. You need to take into consideration one topic at a time, learn it, practice it and implement the algorithm/s in it using a language choice of yours. This is the best way to start studying Machine Learning. Practicing one topic at a time, very soon you would acquire the width that is eventually required of a Machine Learning expert.

6. References

- [1] Aziz S, Mohamed E, Youssef F (2018) Traffic sign recognition based on multi-feature fusion and ELM classifier. *Proc Comput Sci* 127:146–153.
- [2] Jang, C., Kim, H., Park, E., Kim, H. (2016). Data debiased traffic sign recognition using MSERs and CNN. In 2016 International Conference on Electronics, Information, and Communications (ICEIC), Da Nang, Vietnam, pp. 1-4. <https://doi.org/10.1109/ELINFOCOM.2016.7562938>.
- [3] Lai, Y., Wang, N., Yang, Y., & Lin, L. (2018). Traffic signs recognition and classification based on deep feature learning. In 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM), Madeira, Portugal (pp. 622-629).
- [4] Rosario G, Sonderman T, Zhu X. (2018) Deep Transfer Learning for Traffic Sign Recognition[C]//2018IEEE International Conference on Information Reuse and Integration (IRI). IEEE: 178–185. MLA.
- [5] Hatolkar, Y., Agarwal, P., & Patil, S. (2018). A Survey on Road Traffic Sign Recognition System using Convolution Neural Network.
- [6] Huang, Z., Yu, Y., Gu, J., & Liu, H. (2017). An efficient method for traffic sign recognition based on extreme learning machine. *IEEE transactions on cybernetics*, 47(4), 920-933.

- [7] Li, C., Hu, Y., Xiao, L., Tian, L. (2012). Salient traffic sign recognition based on sparse representation of visual perception. In 2012 International Conference on Computer Vision in Remote Sensing, Xiamen, China, pp. 273-278. <https://doi.org/10.1109/CVRS.2012.6421274>.
- [8] Shi W, Xin L, Yu Z et al (2017) An FPGA-based hardware accelerator for traffic sign detection. *IEEE Transactions on Very Large Scale Integration Systems* 4:1362–1372.
- [9] Rosario G, Sonderman T, Zhu X. (2018) Deep Transfer Learning for Traffic signal Recognition[C] 2018 IEEE International Conference on Information Reuse and Integration.
- [10] Li, H., Gong, M. (2017). Self-paced Convolutional Neural Networks. In *IJCAI*, pp. 2110-2116.
- [11] Yang, Y., Luo, H., Xu, H., Wu, F. (2015). Towards real-time traffic sign detection and classification. *IEEE Transactions on Intelligent Transportation Systems* 17(7):2022-2031. <https://doi.org/10.1109/TITS.2015.2482461>
- [12] Yi, K., Jian, Z., Chen, S., Chen, Y., & Zheng, N. (2018). Knowledgebased Recurrent Attentive Neural Network for Traffic Sign Detection. *arXiv preprint arXiv:1803.05263*.
- [13] Promlainak, S.; Woraratpanya, K.; Kuengwong, J.; Kuroki, Y. Thai tra_c sign detection and recognition for driver assistance. In *Proceedings of the 7th ICT International Student Project Conference (ICT - ISPC)*, Nakhon Pathom, Thailand, 11–13 July 2018; pp. 1–5.
- [14] Yuan, Y.; Xiong, Z.T.; Wang, Q. An incremental framework for video based traffic sign detection, tracking, and recognition. *IEEE Trans. Intell. Transp. Syst.* 2017, 18, 1918–1929

7.Source Code:



```
1 from flask import *
2 import os
3 from werkzeug.utils import secure_filename
4 from keras.models import load_model
5 import numpy as np
6 from PIL import Image
7
8 app = Flask(__name__)
9
10 # Classes of traffic signs
11 classes = {0: 'Speed limit (20km/h)',
12            1: 'Speed limit (30km/h)',
13            2: 'Speed limit (50km/h)',
14            3: 'Speed limit (60km/h)',
15            4: 'Speed limit (70km/h)',
16            5: 'Speed limit (80km/h)',
17            6: 'End of speed limit (80km/h)',
18            7: 'Speed limit (100km/h)',
19            8: 'Speed limit (120km/h)',
20            9: 'No passing',
21            10: 'No passing veh over 3.5 tons',
22            11: 'Right-of-way at intersection',
23            12: 'Priority road',
24            13: 'Yield',
25            14: 'Stop',
26            15: 'No vehicles',
27            16: 'Vehicle > 3.5 tons prohibited',
28            17: 'No entry'}
```

Fig-19-source code

ROUTE FUNCTION:

The route() function of the Flask class is a decorator, which tells the application which URL should call the associated function.

app.route(rule, options)

The rule parameter represents URL binding with the function.

The options is a list of parameters to be forwarded to the underlying Rule object.

```
@app.route('/')
def index():
    return render_template('index.html')
```

Fig-19.1-route function code

TESTING CODE:

This is the code for testing the image which is same as we used during validation. Here we load our model and open the image and we resize it and we return the testing data.

```
def image_processing(img):
    model = load_model('./model/TSR.h5')
    data=[]
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
    X_test=np.array(data)
    Y_pred = model.predict_classes(X_test)
    return Y_pred
```

Fig-19.2

Command prompt:

Command Prompt

```
D:\internship\traffic>python -m venv venv/

D:\internship\traffic>source venv/bin/activate
'source' is not recognized as an internal or external command,
operable program or batch file.

D:\internship\traffic>source venv/Scripts/activate
'source' is not recognized as an internal or external command,
operable program or batch file.

D:\internship\traffic>cd venv

D:\internship\traffic\venv>.Scripts\activate
(venv) D:\internship\traffic\venv>pip install flask
Collecting flask
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl (94kB)
    100% |#####| 102kB 763kB/s
Collecting itsdangerous>=0.24 (from flask)
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting Werkzeug>=0.15 (from flask)
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b318f71d62e/Werkzeug-1.0.1-py2.py3-none-any.whl (298kB)
    100% |#####| 307kB 633kB/s
Collecting click>=5.1 (from flask)
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/click-7.1.2-py2.py3-none-any.whl (82kB)
    100% |#####| 92kB 368kB/s
Collecting Jinja2>=2.10.1 (from flask)
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/30/9e/f663a2aa66a09d838042ae1a2c5659828bb9b41ea3a6efa20a20fd92b121/Jinja2-2.11.2-py2.py3-none-any.whl (125kB)
    100% |#####| 133kB 273kB/s
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask)
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/b9/82/833c7714951bff8f502ed054e6fbd8bd00e083d1fd96de6a46905cf23378/MarkupSafe-1.1.1-cp36-cp36m-win_amd64.whl
Installing collected packages: itsdangerous, Werkzeug, click, MarkupSafe, Jinja2, flask
Successfully installed Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 flask-1.1.2 itsdangerous-1.1.0
You are using pip version 9.0.1, however version 20.2.4 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

FLASK CODE:-

from flask import *

import os

```
from werkzeug.utils import secure_filename
```

```
from keras.models import load_model
```

```
import numpy as np
```

```
from PIL import Image
```

```
app = Flask(__name__)
```

```
# Classes of traffic signs
```

```
classes = { 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 veh over 3.5 tons',
```

```
           11:'Right-of-way at intersection',
```

```
           12:'Priority road',
```

```
           13:'Yield',
```

: 'Stop',

: 'No vehicles',

: 'Vehicle > 3.5 tons prohibited',

: 'No entry',

: 'General caution',

: 'Dangerous curve left',

: 'Dangerous curve right',

: 'Double curve',

: 'Bumpy road',

: 'Slippery road',

: 'Road narrows on the right',

: 'Road work',

: 'Traffic signals',

: 'Pedestrians',

: 'Children crossing',

: 'Bicycles crossing',

: 'Beware of ice/snow',

: 'Wild animals crossing',

'End speed + passing limits',

'Turn right ahead',

: 'Turn left ahead',

```
: 'Ahead only',  
  
: 'Go straight or right',  
  
: 'Go straight or left',  
  
: 'Keep right',  
  
: 'Keep left',  
  
: 'Roundabout mandatory',  
  
: 'End of no passing',  
  
: 'End no passing vehicle > 3.5 tons' }
```

```
def image_processing(img):  
  
    model = load_model('./model/TSR.h5')  
  
    data=[]  
  
    image = Image.open(img)  
  
    image = image.resize((30,30))  
  
    data.append(np.array(image))  
  
    X_test=np.array(data)  
  
    Y_pred = model.predict_classes(X_test)  
  
    return Y_pred  
  
  
@app.route('/')  
  
def index():
```

```

return render_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':

        # Get the file from post request

        f = request.files['file']

        file_path = secure_filename(f.filename)

        f.save(file_path)

        # Make prediction

        result = image_processing(file_path)

        s = [str(i) for i in result]

        a = int("".join(s))

        result = "Predicted Traffic 🚦 Sign is: " + classes[a]

        os.remove(file_path)

        return result

    return None

if __name__ == '__main__':

    app.run(debug=True)

```