

# **Traffic Signals Recognition (TSR)**

*Project phase-II report submitted  
in  
partial fulfillment of requirement for the award of degree of*

## **Bachelor of Technology in Information Technology**

*By*

**Mr. Haidar Ahmad**

**Mr. Haidar Hasan**

*Guide*

**Prof. Swati Shamkuwar**



**Department of Information Technology  
G H Raisoni College of Engineering, Nagpur**

(An Autonomous Institute affiliated to Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur)

Accredited by NAAC with “A+” Grade

Ranked 130<sup>th</sup> by NIRF, MHRD in the Engineering Category for India Ranking 2021,  
Ranked 2<sup>nd</sup> by ARIIA 2020, MHRD in Private or Self Finance Institutions, 5 Star Rating  
by MIC, MHRD 2021

## Declaration

We, hereby declare that the project phase-II report titled “**Traffic Signals Recognition (TSR)**” submitted herein has been carried out by us towards partial fulfillment of requirement for the award of Degree of Bachelor of Technology in **Information Technology**. The work is original and has not been submitted earlier as a whole or in part for the award of any degree / diploma at this or any other Institution / University.

We also hereby assign to G H Rasoni College of Engineering, Nagpur all rights under copyright that may exist in and to the above work and any revised or expanded derivatives works based on the work as mentioned. Other work copied from references, manuals etc. are disclaimed.

**Place: Nagpur, Maharashtra**  
**Date**

**Haidar Ahmad**  
**Haidar Hasan**

## Certificate

The project phase-I report entitled as “**Traffic Signals Recognition (TSR)**” submitted by **Haidar Ahmad, Haidar Hasan** for the award of Degree of Bachelor of Technology in Information Technology has been carried out under our supervision. The work is comprehensive, complete and fit for evaluation.



**Institute Guide**  
**Prof. Swati Shamkuwar**  
**Assistant Professor**

Department of Information Technology  
G H R C E, Nagpur

# **RAISONI GROUP**

— a vision beyond —

**Dr. Mahendra Gaikwad**  
**Head**  
Department of Information Technology  
G H R C E, Nagpur

**Dr. Sachin Untawale**  
**Director**  
GHRCE, Nagpur

## **ACKNOWLEDGEMENT**

We would like to take this opportunity to express our deep sense of gratitude to all who helped me directly or indirectly during this work.

We are thankful to our project guide, Prof. Swati Shamkuwar, Department of Information Technology, GHRCE, Nagpur for guiding us through this entire project and giving us good lessons and we wish to gain valuable knowledge and guidance from her in future also.

We are very grateful to Dr. Mahendra Gaikwad, Head of Department of Information Technology GHRCE, Nagpur and Dr. Sachin Untawale Director of GHRCE, Nagpur for his constant support and for providing necessary facilities for conducting out the project. we had a lot of fun completing this project and also learned a lot and we are sure that in the upcoming projects, we will actively participate and complete them with integrity.

Finally, we would like to express our gratitude to our family members who supported and encourages us all the time.

## الإهداء

إلى من شجعني على المثابرة طوال عمري إلى الرجل  
الأبرز في حياتي.....

(والدي العزيز)

إلى من بها ,أعلو وعليها , أرتكز إلى القلب المعطاء.....

(والدتي العزيزة)

من بذلوا جهدا في مساعدتي وكانوا لي سنداً إلى أخوتي.....

(علي/ ميس )

إلى أسرتي إلى أصدقائي وزملائي

إلى كل من ساهم ولو بحرف في حياتي الدراسية

## ABSTRACT

*There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs classification is the process of identifying which class a traffic sign belongs to.*

*In this Python project, we will build 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.*

*Nowadays, more and more object recognition tasks are being solved with Convolutional Neural Networks (CNN). Due to its high recognition rate and fast execution, the convolutional neural networks have enhanced most of computer vision tasks, both existing and new ones. In this project, we propose an implementation of traffic signs recognition algorithm using a convolution neural network. Training of the neural network is implemented using the TensorFlow library.*

*Self-Driving Cars or Autonomous Cars provide many benefits for humanity, such as reduction of deaths and injuries in road accidents, reduction of air pollution, increasing the quality of car control. For this purpose, some cameras or sensors are placed on the car, and an efficient control system must be set up, this system allows to receive images from different cameras and/or sensors in real-time especially those representing traffic signs, and process them to allows high autonomous control and driving of the car. Among the most promising algorithms used in this field, we find convolutional neural networks CNN. In the present work, we have proposed a CNN model. As programming tools, we have used python, Tensorflow, and Keras which are currently the most used in the field.*

# CONTENTS

Abstract.....	I
Contents .....	II
List of Figures .....	IV
Introduction.....	1
<b>Problem Summary and Introduction</b> .....	1
<b>Aim and Objectives</b> .....	1
<b>Problem Specifications</b> .....	2
<b>Literature Review and Prior Art Search (PAS)</b> .....	2
<b>Materials / Tools required</b> .....	4
Analysis, Design Methodology and Implementation Strategy .....	5
<b>Observation Matrix</b> .....	5
<b>Ideation Canvas</b> .....	6
<b>Product Development Canvas</b> .....	7
<b>Dataset</b> .....	8
<b>System Design</b> .....	10
Implementation .....	15
<b>Implemented Functionality</b> .....	15
<b>pre-Requested</b> .....	15
Tensorflow .....	15
Keras .....	16
Pillow .....	20
Numpy .....	22
Tqdm.....	24
Jupyter Notebook.....	25
VS code.....	27
<b>Project File Structure</b> .....	28
<b>Getting and Performing Data Cleaning</b> .....	29
<b>Building the Model</b> .....	31
<b>Training the model</b> .....	32
<b>Results and Reports</b> .....	35
<b>Snapshots</b> .....	36
<b>Testing and Verification</b> .....	39
Conclusion .....	41
Scope of future work .....	41
References.....	42





## LIST OF FIGURES

<i>Figure 1</i> Observation Matrix	6
<i>Figure 2</i> Ideation Canvas	7
<i>Figure 3</i> Product Development Canvas	8
<i>Figure 4</i> Dataset main folder	9
<i>Figure 5</i> example of sub-folder	10
<i>Figure 6</i> Training Dataset folder	10
<i>Figure 7</i> System Block Diagram	11
<i>Figure 8</i> Activity Diagram	12
<i>Figure 9</i> Sequence Diagram	13
<i>Figure 10</i> Class Diagram	13
<i>Figure 11</i> Data flow Diagram	14
<i>Figure 12</i> TensorFlow	16
<i>Figure 13</i> Keras	20
<i>Figure 14</i> NumPy	24
<i>Figure 15</i> Jupyter Notebook	27
<i>Figure 16</i> VS code	28
<i>Figure 17</i> Project File Structure	28
<i>Figure 18</i> performing Data cleaning	31
<i>Figure 19</i> performing Data cleaning	31
<i>Figure 20</i> Loading the model	32
<i>Figure 21</i> Batch example	34
<i>Figure 22</i> Training the model	35
<i>Figure 23</i> building the GUI	36
<i>Figure 24</i> building the GUI	36
<i>Figure 25</i> Importing the libraries	36
<i>Figure 26</i> Training steps	37
<i>Figure 27</i> TSR GUI	37
<i>Figure 28</i> Uploading an image to GUI	38
<i>Figure 29</i> Classify the image	38
<i>Figure 30</i> Accuracy diagram of the model	39
<i>Figure 31</i> Data loss diagram	40

# INTRODUCTION

## Problem Summary and Introduction

Traffic-sign recognition (TSR) has been an essential part of driver-assistance systems, which is able to assist drivers in avoiding a vast number of potential hazards and improve the experience of driving. However, the TSR is a realistic task that is full of constraints, such as visual environment, physical damages, and partial occasions, etc. In order to deal with such constraints, convolutional neural networks (CNN) are widely used to extract the features of traffic signs and classify them into corresponding classes.

In this project, we will be using The German Traffic Sign Benchmark, which is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011.

## Aim and Objectives

Traffic scene understanding is an important topic in the field of computer vision and intelligent systems. Traffic signs effectively assist drivers in the process of driving and keep them drive more safely as they are designed to inform drivers of current road situations and potential hazards. These signs are normally rigid and simple shapes, such as circles, triangles and regular polygons, and with eye-catching colors.

Hence, traffic sign recognition has become more and more valuable for driver-assistance systems, highway maintenance and especially for self-driving vehicles.

In the development of traffic sign recognition, German traffic-sign detection and classification benchmarks brought in a vast majority of benefits for evaluation across various algorithms, which were not comparable until the release of the benchmarks. German Traffic Sign Detection Benchmark (GTSDB) and German Traffic Sign Recognition Benchmark (GTSRB) presented two public extensive and available datasets and nowadays there are several methods have achieved high accuracy rate based on these datasets. Besides, other datasets are also available in public recent years, such as LISA traffic sign dataset (LISATSD), Swedish Traffic Signs Dataset (STSD) and Chinese Traffic

Sign Dataset (CTSD). The GTSRB and GTSDb datasets are the most popular ones for recognizing traffic signs and many methods have achieved great success on them.

Overall, the ultimate objective of this report is to complete the customized traffic-sign recognition and figure out which state-of-art networks better fit into this project.

## **Problem Specifications**

Generally, there are two steps in a typical traffic sign recognition. The first one is to locate and get size information of traffic signs in natural scene images, which is known as traffic sign detection. The second step is to categorize detected traffic signs into the corresponding sub-classes, which is known as traffic sign classification, and this step is generally completed manually. Although traffic sign recognition has gained a plethora of popularity in driver assistant system, there are still numerous difficulties for identifying real-world traffic signs by using computer algorithms due to various size of targets, color deterioration and partial occlusion.

In order to deal with these obstacles, many approaches and algorithms have been proposed. In the past, traffic sign detection mainly relied on traditional object detection algorithms and the pipeline of traffic sign detection normally utilized hand-crafted features to extract region proposals, and then combined classifiers to filter out the negatives. Recently, deep learning methods are emerging, and various cutting-edge approaches have been widely applied into this area, such as deep convolutional networks (CNNs). CNNs have brought possibility of learning features from giant amount of data without preprocessing, which avoids the process of designing hand-crafted features and absorbs more generalized features. Besides, CNN has been already set forth as an object classifier in machine learning which have been leveraged on traffic sign classification.

## **Literature Review and Prior Art Search (PAS)**

In Background work the various reference of the existing projects are taken into consideration which are similar to this current project.

Traffic sign detection and recognition using fully convolutional network guided proposals [1]

Detecting and recognizing traffic signs is a hot topic in the field of computer vision with lots of applications, e.g., safe driving, path planning, robot navigation etc. We propose a

novel framework with two deep learning components including fully convolutional network (FCN) guided traffic sign proposals and deep convolutional neural network (CNN) for object classification. Our core idea is to use CNN to classify traffic sign proposals to perform fast and accurate traffic sign detection and recognition. Due to the complexity of the traffic scene, we improve the state-of-the-art object proposal method, Edge Box, by incorporating with a trained FCN. The FCN guided object proposals can produce more discriminative candidates, which help to make the whole detection system fast and accurate. In the experiments, we have evaluated the proposed method on publicly available traffic sign benchmark, Swedish Traffic Signs Dataset (STSD), and achieved the state-of-the-art results.

Malaysia traffic sign recognition with convolutional neural network [2]

Traffic sign recognition system is an important subsystem in advanced driver assistance systems (ADAS) that assisting a driver to detect a critical driving scenario and subsequently making an immediate decision. Recently, deep architecture neural network is popular because it adapts well in various kind of scenarios, even those which were not used during training. Therefore, a deep architecture neural network is implemented to perform traffic sign classification in order to improve the traffic sign recognition rate. A comparative study for a deep and shallow architecture neural network is presented in this paper. Deep and shallow architecture neural network refer to convolutional neural network (CNN) and radial basis function neural network (RBFNN) respectively. In the simulation result, two types of training modes had been compared i.e. incremental training and batch training. Experimental results show that incremental training mode trains faster than batch training mode. The performance of the convolutional neural network is evaluated with the Malaysian traffic sign database and achieves 99% of the recognition rate.

Traffic Sign Detection and Recognition using a CNN Ensemble [3]

In today's world, almost everything we do has been simplified by automated tasks. In an attempt to focus on the road while driving, drivers often miss out on signs on the side of the road, which could be dangerous for them and for the people around them. This problem can be avoided if there was an efficient way to notify the driver without having them to shift their focus. Traffic Sign Detection and Recognition (TSDR) plays an important role here by detecting and recognizing a sign, thus notifying the driver of any upcoming signs. This not only ensures road safety, but also allows the driver to be at little more ease while driving on tricky or new roads. Another commonly faced problem is not being able to understand the meaning of the sign. With the help of this Advanced Driver Assistance

Systems (ADAS) application, drivers will no longer face the problem of understanding what the sign says. In this paper, we propose a method for Traffic Sign Detection and Recognition using image processing for the detection of a sign and an ensemble of Convolutional Neural Networks (CNN) for the recognition of the sign. CNNs have a high recognition rate, thus making it desirable to use for implementing various computer vision tasks. TensorFlow is used for the implementation of the CNN. We have achieved higher than 99% recognition accuracies for circular signs on the Belgium and German data sets.

## **Materials / Tools required**

Software requirements:

- ❖ OS: Windows 7 or above, Recommended: Windows 10.
- ❖ For development: Python 3.9 or above, Jupyter Notebook, Anaconda 3 or above, reliable internet bandwidth.
- ❖ For Execution: Python 3.9 and above, Command Prompt.

Hardware requirements:

- ❖ CPU: A minimum of 8th generation (Intel Core i5 processor) 2.30 GHz is recommended.
- ❖ RAM: A minimum of 8 GB RAM.
- ❖ Storage: A minimum of 20 GB.

# **ANALYSIS, DESIGN METHODOLOGY AND IMPLEMENTATION STRATEGY**

## **Observation Matrix**

Observation Matrix (figure – 1) is a way to find out how dependent the measurements are on the condition of the system.

The matrix consists of 4 parts:

Observations, which includes all the possible platform we can use to introduce the model, each one of them has its own pros and cons, although many of them are viable options to develop the mode, as a starting point we have to begin development on the most suitable platform, the observations are:

- ❖ The website
- ❖ Web Application
- ❖ Android Application
- ❖ Linux App
- ❖ Mac program
- ❖ Windows Software
- ❖ IOS APP

The scouted challenges, after careful consideration and research, the option to develop on an IOS app was not viable considering the limited user base so it was eliminated.

Based on the desirability, feasibility, Viability, it became clear that having web application is not suitable seeing as there are better options.

Lastly, seeing as the most popular method was the Windows Software, it was considered as the desired method to develop the model with, furthermore, the development process will take less running time and help the developer in being more efficient in his work.

## Observation Matrix

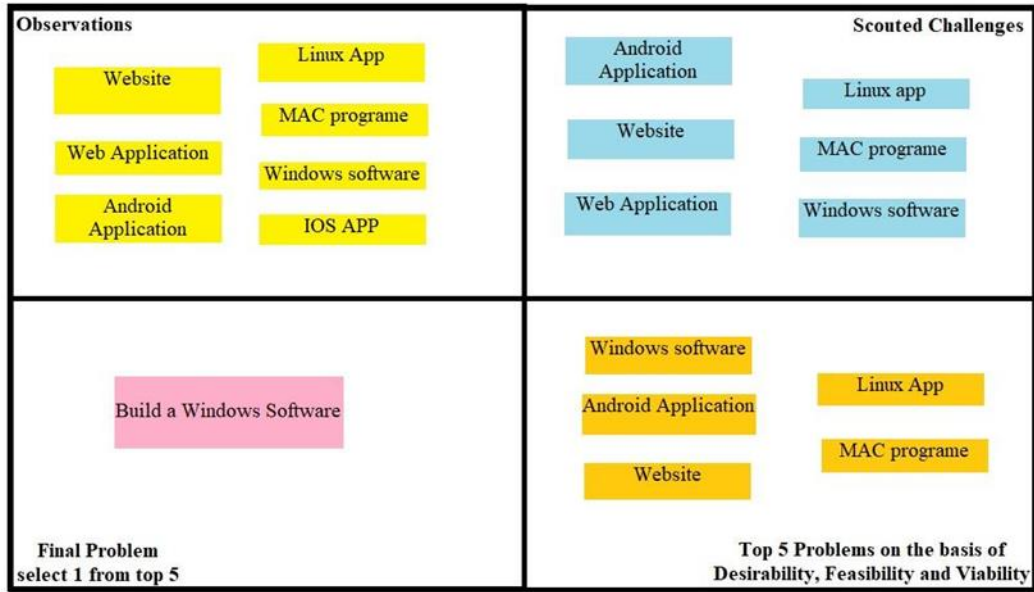


Figure 1 Observation Matrix

## Ideation Canvas

The Ideation canvas will help to make the ideas clear in order to implement them into the product. First the people are defined, Engineers, Drivers. By defining the people, we are able to better understand the features that need to be implemented and the requirements.

The activities of the model only include, detecting the traffic sign from a photo, and describe it.

Thirdly, the Situation/Context/Location where the product will be used, can be defined as follows: The product can be hosted by a website using cloud computing services which allows for better reach and less requirement needed to run it since websites use SAAS (Software as A Service) model. Also, the Product can be introduced as a mobile application, which will help make it available to a wider range of users and make it accessible from different environments.

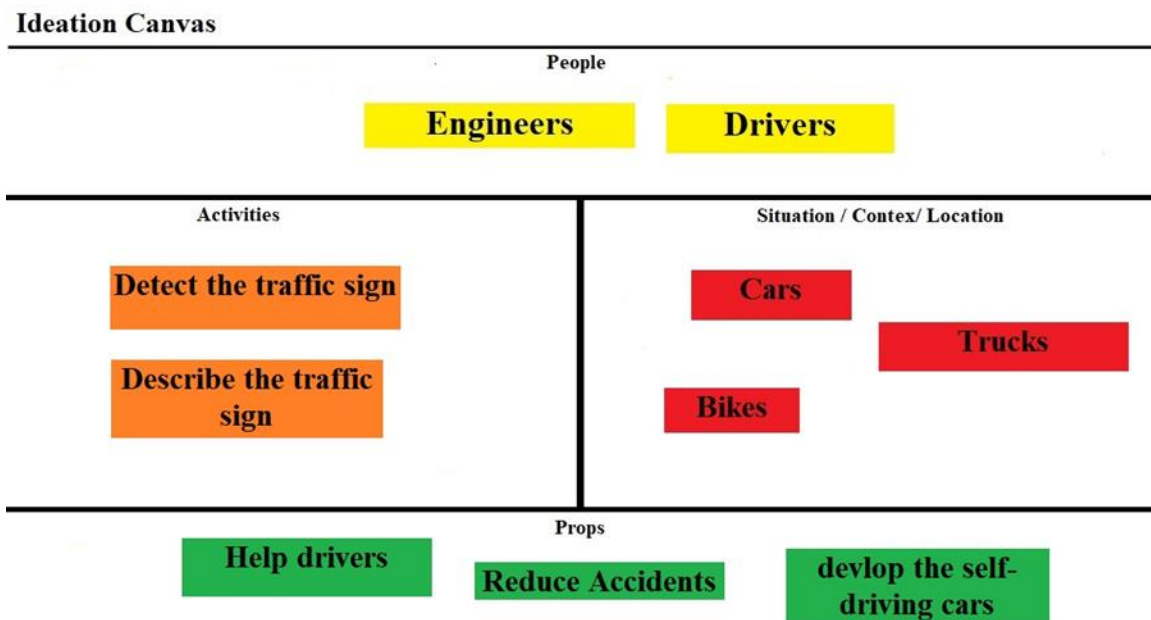


Figure 2 Ideation Canvas

## Product Development Canvas

Product Development Canvas is a planning canvas intended to build products that have a good user experience through an aim to work on feature development. It includes both agile methodologies with UX principles to aid the validation of the product solutions. In this canvas, there are 8 components. The first one, Purpose, explains how the goal is to produce an accurate satisfying description which is accomplished by constantly optimizing the algorithms and feeding the model with the proper datasets. Secondly comes the people, referring to who can use the product, which are all kind of users who require such technology without any specific category of users. It can be a regular person, like Driver, or any software Developer. Next comes the product experience, which after a brief survey the experience was described as Easy, Simple, and doesn't require any technical knowledge or specific skill in any domain. Then comes a very important part, the product functions, which can be explained as follows, the product takes in an input from the user, it then performs analysis on the input by utilizing neural networks, which helps detect the traffic signal. The product features are what makes it stand out, its user-friendly GUI makes it easy for users to navigate its functionality and get a good user experience, secondly, its computer application and wide spectrum of possibilities makes it desirable and popular. The components for this product are easy to attain and simple. First, a pc is highly recommended to host to product and use it, although the specifications of the pc are high,



users can use the product without any performance issues. The customer revalidation, the revalidation by the customer was that the product was easy to use and didn't require much time to navigate through, secondly, it's multipurpose and wide range of use cases was highly applauded. The reject/reassign/retain, the product received some negative feedback, such as requiring high end device in order to run it on, the high amount of data needed to handle was also an issue that needed to be addressed which leads to the next point. Long training time is issue that needs to be worked on, less amount of data and faster algorithm needs to be introduced and improved.

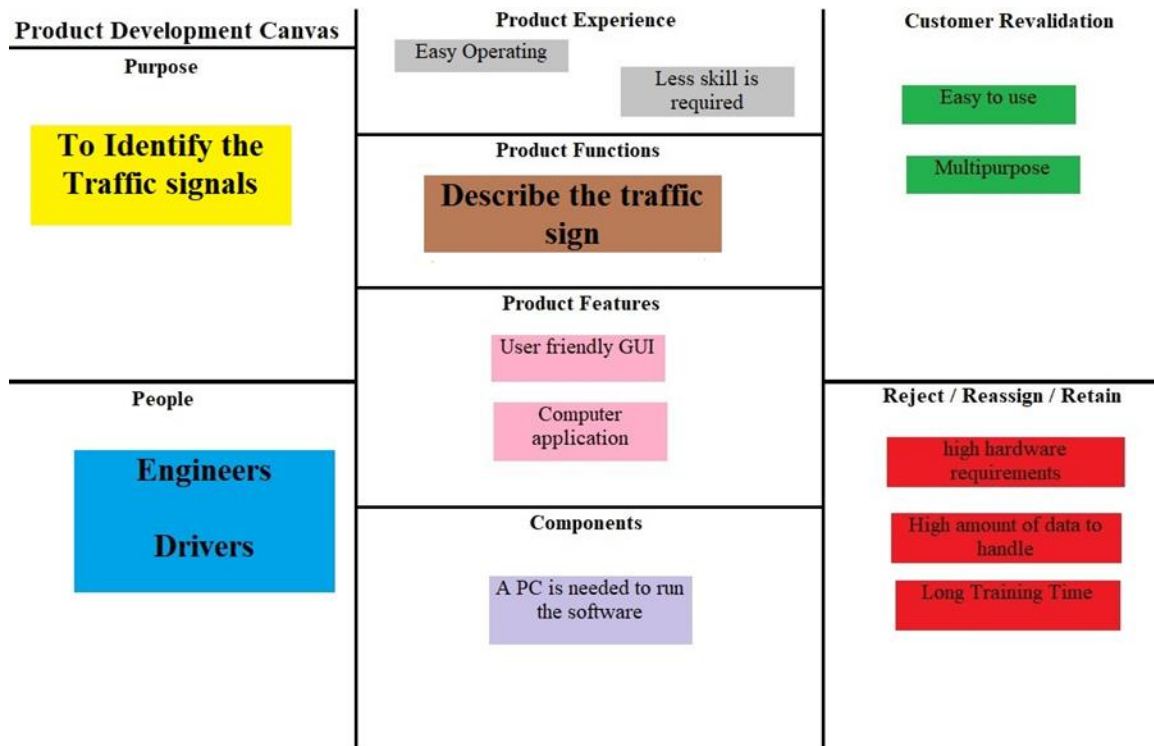


Figure 3 Product Development Canvas

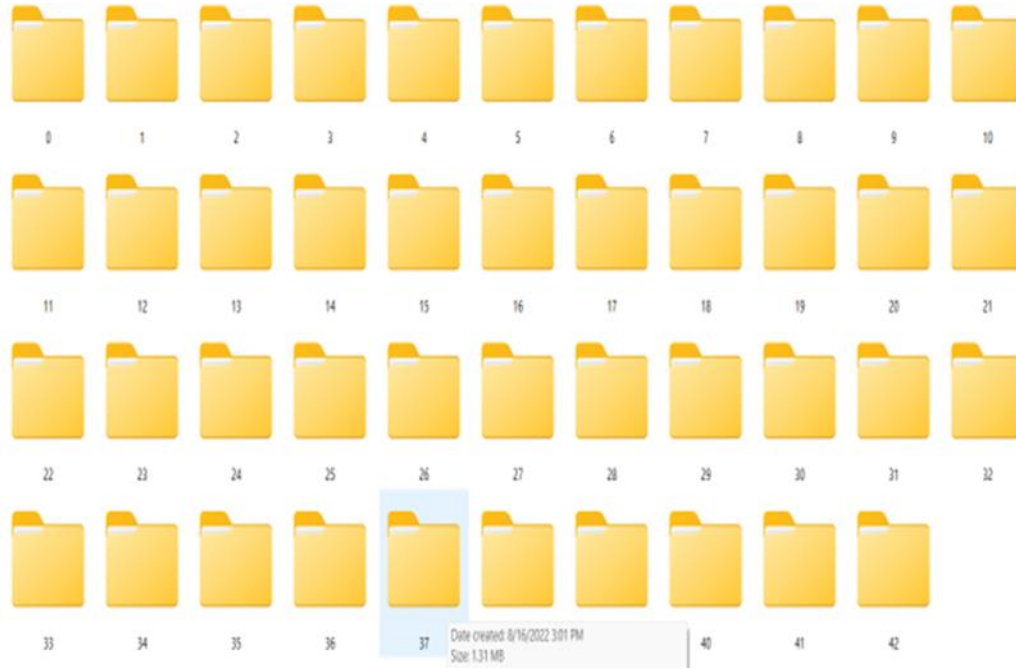
## Dataset

We will be using a data set consists of 51840 photos divided into 39209 photos for training the model and 12631 photo for testing the model

The training data set is divided into 43 classes as follows:

(1:Speed limit (20km/h), 2:Speed limit (30km/h), 3:Speed limit (50km/h), 4:Speed limit (60km/h), 5:Speed limit (70km/h), 6:Speed limit (80km/h), 7:End of speed limit (80km/h), 8:Speed limit (100km/h), 9:Speed limit (120km/h), 10:No passing, 11:No passing vehicle over 3.5 tons, 12:Right-of-way at intersection, 13:Priority road,14:Yield, 15:Stop,16:No

vehicles, 17:Veh > 3.5 tons prohibited, 18:No entry,19:General caution, 20:Dangerous curve left, 21:Dangerous curve right, 22:Double curve, 23:Bumpy road, 24:Slippery road, 25:Road narrows on the right, 26:Road work, 27:Traffic signals, 28:Pedestrians, 29:Children crossing, 30:Bicycles crossing, 31:Beware of ice/snow, 32:Wild animals crossing, 33:End speed + passing limits, 34:Turn right ahead, 35:Turn left ahead, 36:Ahead only, 37:Go straight or right, 38:Go straight or left, 39:Keep right, 40:Keep left 41:'Roundabout mandatory', 42:End of no passing,43:End no passing veh > 3.5 tons)



*Figure 4 Dataset main folder*

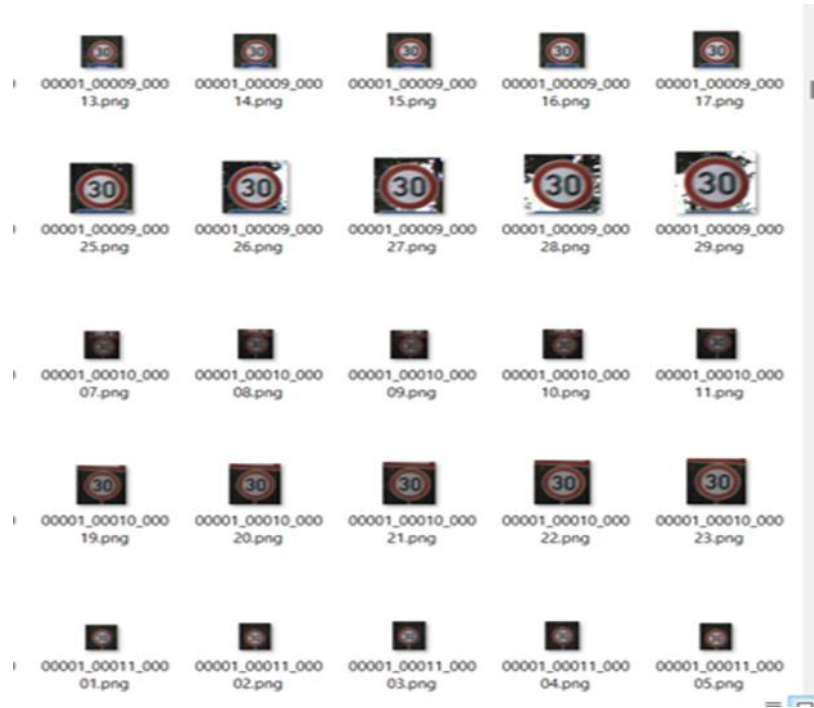


Figure 5 example of sub-folder

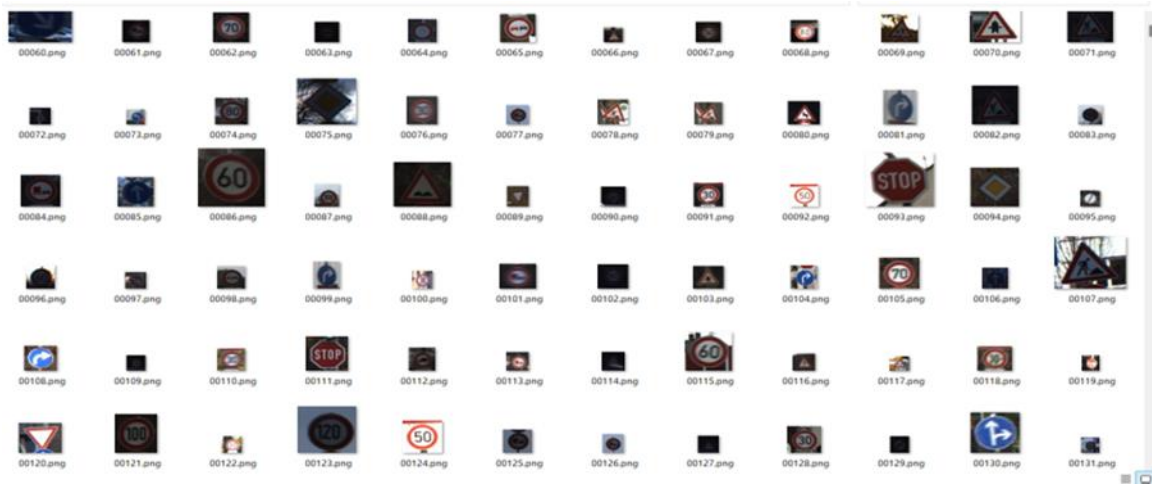


Figure 6 Training Dataset folder

## System Design

In this Python project, we will build a deep neural network model that can classify traffic signs present in the image into different categories. With this model, we will be able to read and understand traffic signs which are a very important task for all autonomous vehicles.

Our approach to building this traffic sign classification model is discussed as:

- ❖ Explore the dataset

- ❖ Build a CNN model
- ❖ Train and validate the model
- ❖ Test the model with test dataset

The model we used is built with Keras and Tensorflow using Convolutional Neural Networks (CNN). A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple layers. A convolution operation is performed on these layers using a filter that performs 2D matrix multiplication on the layer and filter.

*Figure 7 System Block Diagram*

Activity diagram is one of the most important diagrams in UML to show the dynamic traits of the model, Activity diagram is simply a diagram to represent the flow from one activity to another activity. The activity can be shown as an operation of the model.

The main goal of activity diagrams is almost the same to the other four diagrams. It highlights the dynamic interactions of the system. Other four diagrams are used to illustrate the message flow from one object to another but this diagram is used to show message flow from one activity to another.

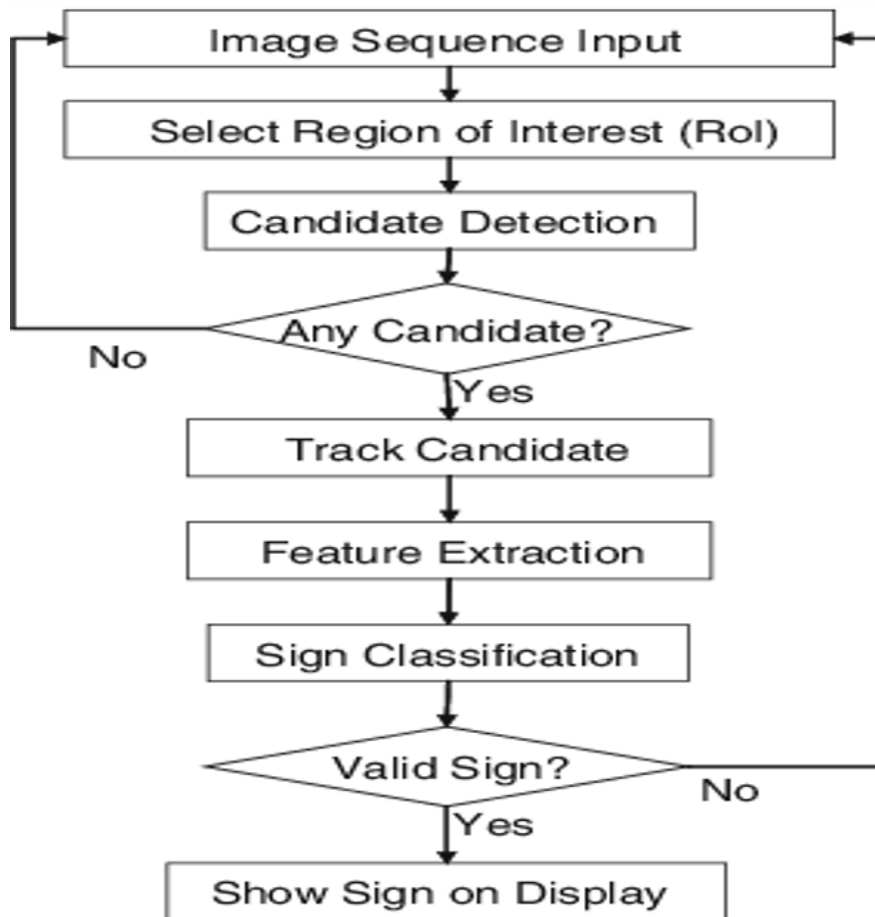


Figure 8 Activity Diagram

Sequence Diagram in explains how the sequence of the program is, to briefly summarize it.

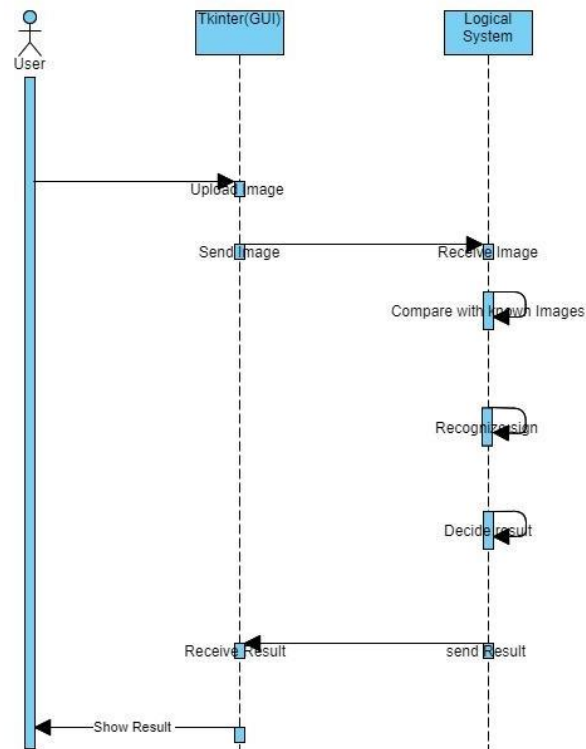


Figure 9 Sequence Diagram

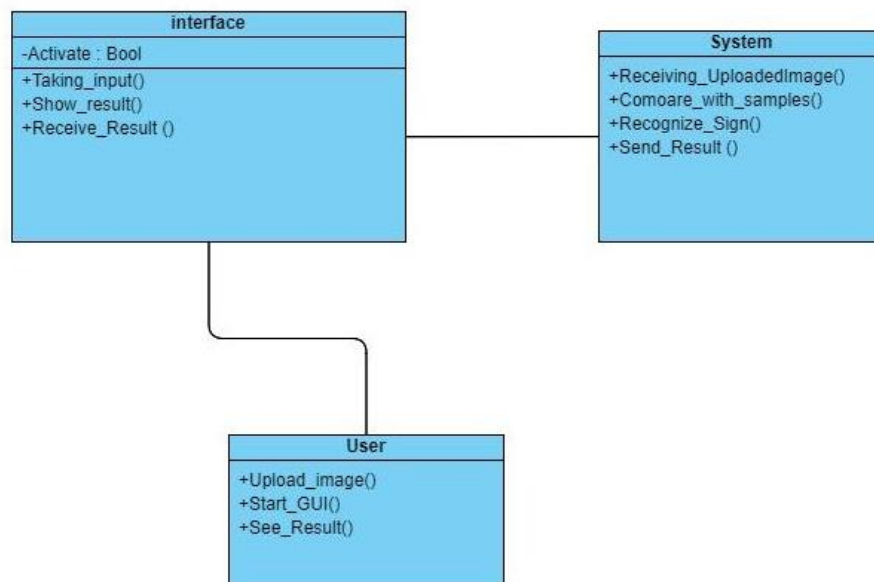


Figure 10 Class Diagram

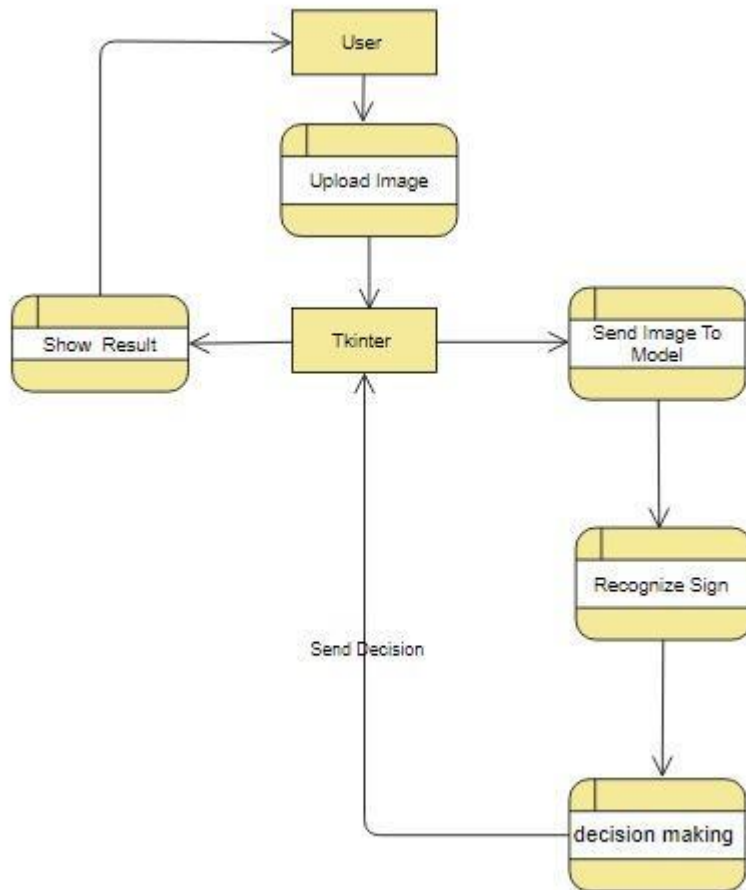


Figure 11 Data flow Diagram

# IMPLEMENTATION

## Implemented Functionality

### pre-Requested

This project requires good knowledge of Deep learning, Python, working on Jupyter notebooks, Keras library, Numpy, and Natural language processing. We have installed all the following necessary libraries:

- ❖ Tensorflow
- ❖ Keras
- ❖ Pillow
- ❖ Numpy
- ❖ Tqdm
- ❖ Jupyter Notebook
- ❖ VS code

### Tensorflow

TensorFlow is a Python library for fast numerical computing created and released by Google. 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 TensorFlow. TensorFlow is an open-source library for fast numerical computing.

It was created and is maintained by Google and was released under the Apache 2.0 open-source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least of which is RankBrain in Google search and the fun DeepDream project. It can run on single CPU systems and GPUs, as well as mobile devices and large-scale distributed systems of hundreds of machines.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Let us now consider the following important features of TensorFlow



- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

TensorFlow is well-documented and includes plenty of machine learning libraries. It offers a few important functionalities and methods for the same.

TensorFlow is also called a “Google” product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.



*Figure 12 TensorFlow*

## **Keras**

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is:

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.

- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

## Keras & TensorFlow 2

TensorFlow 2 is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities:

- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination. It cannot handle low-level computations, so it makes use of the Backend library to resolve it. The backend library act as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.

Initially, it had over 4800 contributors during its launch, which now has gone up to 250,000 developers. It has a 2X growth ever since every year it has grown. Big companies like Microsoft, Google, NVIDIA, and Amazon have actively contributed to the development of Keras. It has an amazing industry interaction, and it is used in the development of popular firms likes Netflix, Uber, Google, Expedia, etc.

What makes Keras special?

- Focus on user experience has always been a major part of Keras.
- Large adoption in the industry.
- It is a multi backend and supports multi-platform, which helps all the encoders come together for coding.
- Research community present for Keras works amazingly with the production community.
- Easy to grasp all concepts.
- It supports fast prototyping.
- It seamlessly runs on CPU as well as GPU.
- It provides the freedom to design any architecture, which then later is utilized as an API for the project.
- It is really very simple to get started with.
- Easy production of models actually makes Keras special.

#### Keras user experience

- Keras is an API designed for humans
- Best practices are followed by Keras to decrease cognitive load, ensures that the models are consistent, and the corresponding APIs are simple.
- Not designed for machines
- Keras provides clear feedback upon the occurrence of any error that minimizes the number of user actions for the majority of the common use cases.
- Easy to learn and use.
- Highly Flexible
- Keras provide high flexibility to all of its developers by integrating low-level deep learning languages such as TensorFlow or Theano, which ensures that anything written in the base language can be implemented in Keras.

#### How Keras support the claim of being multi-backend and multi-platform?

Keras can be developed in R as well as Python, such that the code can be run with TensorFlow, Theano, CNTK, or MXNet as per the requirement. Keras can be run on CPU, NVIDIA GPU, AMD GPU, TPU, etc. It ensures that producing models with Keras is really simple as it totally supports to run with TensorFlow serving, GPU acceleration (WebKeras, Keras.js), Android (TF, TF Lite), iOS (Native CoreML) and Raspberry Pi.

#### Keras Backend

Keras being a model-level library helps in developing deep learning models by offering high-level building blocks. All the low-level computations such as products of Tensor, convolutions, etc. are not handled by Keras itself, rather they depend on a specialized tensor manipulation library that is well optimized to serve as a backend engine. Keras has managed it so perfectly that instead of incorporating one single library of tensor and performing operations related to that particular library, it offers plugging of different backend engines into Keras.

Keras consist of three backend engines, which are as follows:

- TensorFlow

TensorFlow is a Google product, which is one of the most famous deep learning tools widely used in the research area of machine learning and deep neural network. It came into the market on 9th November 2015 under the Apache License 2.0. It is built in such a way that it can easily run on multiple CPUs and GPUs as well as on mobile operating systems. It consists of various wrappers in distinct languages such as Java, C++, or Python.

- Theano

Theano was developed at the University of Montreal, Quebec, Canada, by the MILA group. It is an open-source python library that is widely used for performing mathematical operations on multi-dimensional arrays by incorporating scipy and numpy. It utilizes GPUs for faster computation and efficiently computes the gradients by building symbolic graphs automatically. It has come out to be very suitable for unstable expressions, as it first observes them numerically and then computes them with more stable algorithms.

- CNTK

Microsoft Cognitive Toolkit is deep learning's open-source framework. It consists of all the basic building blocks, which are required to form a neural network. The models are trained using C++ or Python, but it incorporates C# or Java to load the model for making predictions.

Advantages of Keras

Keras encompasses the following advantages, which are as follows:

- It is very easy to understand and incorporate the faster deployment of network models.
- It has huge community support in the market as most of the AI companies are keen on using it.

- It supports multi backend, which means you can use any one of them among TensorFlow, CNTK, and Theano with Keras as a backend according to your requirement.
- Since it has an easy deployment, it also holds support for cross-platform. Following are the devices on which Keras can be deployed:
  - iOS with CoreML
  - Android with TensorFlow Android
  - Web browser with .js support
  - Cloud engine
  - Raspberry pi
- It supports Data parallelism, which means Keras can be trained on multiple GPU's at an instance for speeding up the training time and processing a huge amount of data.

#### Disadvantages of Keras

The only disadvantage is that Keras has its own pre-configured layers, and if you want to create an abstract layer, it won't let you because it cannot handle low-level APIs. It only supports high-level API running on the top of the backend engine (TensorFlow, Theano, and CNTK).



*Figure 13 Keras*

#### **Pillow**

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large

number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders. Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn't support python 3.

Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety of images such as "jpeg", "png", "bmp", "gif", "ppm", "tiff". You can do almost anything on digital images using pillow module. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

### Image Archives

The Python Imaging Library is best suited for image archival and batch processing applications. Python pillow package can be used for creating thumbnails, converting from one format to another and print images, etc.

### Image Display

You can display images using Tk PhotoImage, BitmapImage and Windows DIB interface, which can be used with PythonWin and other Windows-based toolkits and many other Graphical User Interface (GUI) toolkits.

For debugging purposes, there is a show () method to save the image to disk which calls the external display utility.

### Image Processing

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

### What is Image Processing?

Image processing is a method of analyzing and manipulating digital images. The main goal is to extracting some information that can be used somewhere else. Let's make it simple to understand with a real-life example.

For example - Nowadays, CCTV cameras are instated in the traffic light that capture digital images, identify the vehicle's number plate, and check whether the vehicle violated the traffic rules or not. All this process is related to image processing.

## **NumPy**

What is NumPy?

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

NumPy stands for Numerical Python.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities
- Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

Travis Oliphant created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.

It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed.

NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.

The need of NumPy

With the revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.

NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

There are the following advantages of using NumPy for data analysis.

- NumPy performs array-oriented computing.
- It efficiently implements the multidimensional arrays.
- It performs scientific computations.
- It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
- NumPy provides the in-built functions for linear algebra and random number generation.

Nowadays, NumPy in combination with SciPy and Matplotlib is used as the replacement to MATLAB as Python is more complete and easier programming language than MATLAB.

Why is NumPy Faster Than Lists?

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

Which Language is NumPy written in?

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.



*Figure 14 NumPy*



## Tqdm

tqdm is a library in Python which is used for creating Progress Meters or Progress Bars. tqdm got its name from the Arabic name taqaddum which means 'progress'.

Implementing tqdm can be done effortlessly in our loops, functions or even Pandas. Progress bars are pretty useful in Python because:

- One can see if the Kernel is still working.
- Progress Bars are visually appealing to the eyes.
- It gives Code Execution Time and Estimated Time for the code to complete which would help while working on huge datasets

Progress bars are filled up according to the percentage of progress made in accomplishing a task. The progress can be calculated by dividing `number_of_item_processed` by `total_input_item`. Various factors affect the progress bar, such as network speed, latency, and if persisting data into local storage to derive a more accurate ETA (Estimated Time of Arrival).

We can create simple and hassle-free progress bars using the Python external library named tqdm. We can add it to the code and make it look lovely.

The tqdm stands for taqadum in Arabic, which means progress. Python tqdm module works on various platform such Linux, Window, Mac, etc. and it is also compatible with the IPython/Jupyter notebooks.

Need for Progress Bar?

If we are working with the smaller data set, the progress will not bother in our workflow. However, the progress bar can be used for iterating over a dataset, training a model, or encoding a large information set.

- The progress bar provides us an estimation of the process that has been given the approximation of the time it might take more.
- It gives us information that the progress is still running and has not been terminated rudely.

## Jupyter Notebook

Jupyter Notebook is an open-source, web-based interactive environment, which allows you to create and share documents that contain live code, mathematical equations, graphics, maps, plots, visualizations, and narrative text. It integrates with many programming languages like Python, PHP, R, C#, etc.

Notebook documents (or “notebooks”, all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis.

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down.

The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

#### Advantages of Jupyter Notebook

There are the following advantages of Jupyter Notebook

- All in one place: As you know, Jupyter Notebook is an open-source web-based interactive environment that combines code, text, images, videos, mathematical equations, plots, maps, graphical user interface and widgets to a single document.
- Easy to convert: Jupyter Notebook allows users to convert the notebooks into other formats such as HTML and PDF. It also uses online tools and nbviewer which allows you to render a publicly available notebook in the browser directly.

- Easy to share: Jupyter Notebooks are saved in the structured text files (JSON format), which makes them easily shareable.
- Language independent: Jupyter Notebook is platform-independent because it is represented as JSON (JavaScript Object Notation) format, which is a language-independent, text-based file format. Another reason is that the notebook can be processed by any programming language, and can be converted to any file formats such as Markdown, HTML, PDF, and others.
- Interactive code: Jupyter notebook uses ipywidgets packages, which provide many common user interfaces for exploring code and data interactivity.

#### Disadvantages of Jupyter Notebook

There are the following disadvantages of Jupyter Notebook:

- It is very hard to test long asynchronous tasks.
- Less Security
- It runs cell out of order
- In Jupyter notebook, there is no IDE integration, no linting, and no code-style correction.



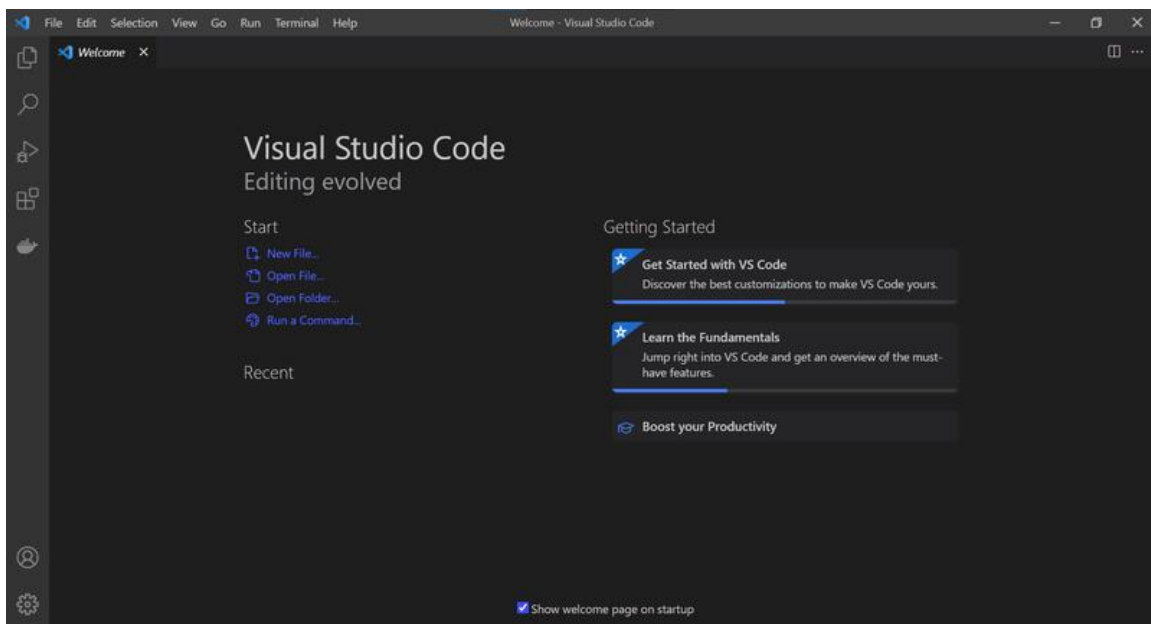
*Figure 15 Jupyter Notebook*

#### **VS code**

Visual Studio Code (famously known as VS Code) is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

VS Code supports a wide array of programming languages from Java, C++, and Python to CSS, Go, and Dockerfile. Moreover, VS Code allows you to add on and even creating new extensions including code linters, debuggers, and cloud and web development support. The VS Code user interface allows for a lot of interaction compared to other text editors. To simplify user experience, VS Code is divided into five main regions:

- The activity bar
- The side bar
- Editor groups
- The panel
- The status bar









*Figure 16 VS code*

## Project File Structure

- ❖ Test folder: Which will contain the Testing dataset.
- ❖ Train folder: Which will contain the Training dataset.
- ❖ Test.csv

The below files will be created by us while making the project.

- ❖ My\_model.h5– this will be our model after training an saving.
- ❖ Traffic Signs.ipynb – Jupyter notebook in which we train and build our Traffic Signals Recognition (TSR).
- ❖ gui.py - Python file for Graphical User Interface

 Test	File folder
 Train	File folder
 gui.py	Python Source File
 my_model.h5	H5 File
 Test.csv	Microsoft Excel Comma S...
 Traffic Signs.ipynb	Jupyter Source File

*Figure 17 Project File Structure*

## Getting and Performing Data Cleaning

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, the success or failure of a project relies on proper data cleaning. Professional data scientists usually invest a very large portion of their time in this step because of the belief that "Better data beats fancier algorithms".

If we have a well-cleaned dataset, there are chances that we can get achieve good results with simple algorithms also, which can prove very beneficial at times especially in terms of computation when the dataset size is large.

Obviously, different types of data will require different types of cleaning.

Steps involved in Data Cleaning:

- Removal of unwanted observations

This includes deleting duplicate/ redundant or irrelevant values from your dataset. Duplicate observations most frequently arise during data collection and Irrelevant observations are those that don't actually fit the specific problem that you're trying to solve.

- Redundant observations alter the efficiency by a great extent as the data repeats and may add towards the correct side or towards the incorrect side, thereby producing unfaithful results.
- Irrelevant observations are any type of data that is of no use to us and can be removed directly.

- Fixing Structural errors

The errors that arise during measurement, transfer of data, or other similar situations are called structural errors. Structural errors include typos in the name of features, the same attribute with a different name, mislabeled classes, i.e. separate classes that should really be the same, or inconsistent capitalization.

For example, the model will treat America and America as different classes or values, though they represent the same value or red, yellow, and red-yellow as different classes or attributes, though one class can be included in the other two classes. So, these are some structural errors that make our model inefficient and give poor quality results.

- Managing Unwanted outliers

Outliers can cause problems with certain types of models. For example, linear regression models are less robust to outliers than decision tree models. Generally, we should not remove outliers until we have a legitimate reason to remove them. Sometimes, removing them improves performance, sometimes not. So, one must have a good reason to remove the outlier, such as suspicious measurements that are unlikely to be part of real data.

- Handling missing data

Missing data is a deceptively tricky issue in machine learning. We cannot just ignore or remove the missing observation. They must be handled carefully as they can be an indication of something important. The two most common ways to deal with missing data are:

- ❖ Dropping observations with missing values.

The fact that the value was missing may be informative in itself.

Plus, in the real world, you often need to make predictions on new data even if some of the features are missing!

- ❖ Imputing the missing values from past observations.

Again, “missingness” is almost always informative in itself, and you should tell your algorithm if a value was missing.

Even if you build a model to impute your values, you’re not adding any real information. You’re just reinforcing the patterns already provided by other features.

Missing data is like missing a puzzle piece. If you drop it, that’s like pretending the puzzle slot isn’t there. If you impute it, that’s like trying to squeeze in a piece from somewhere else in the puzzle.

So, missing data is always an informative and an indication of something important. And we must be aware of our algorithm of missing data by flagging it. By using this technique

of flagging and filling, you are essentially allowing the algorithm to estimate the optimal constant for missingness, instead of just filling it in with the mean.

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

In [2]: data = []
labels = []
classes = 43
cur_path = os.getcwd()

In [3]: for i in range(classes):
path = os.path.join(cur_path, 'train', str(i))
images = os.listdir(path)

for a in images:
try:
image = Image.open(path + '\\' + a)
image = image.resize((30,30))
image = np.array(image)
data.append(image)
labels.append(i)
```

*Figure 18 performing Data cleaning*

```
In [4]: plt.imshow(image, cmap="gray")
plt.show()

In [5]: data = np.array(data)
labels = np.array(labels)

In [6]: print(data.shape, labels.shape)
(39209, 30, 30, 3) (39209,)

In [7]: X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

*Figure 19 performing Data cleaning*

## Building the Model

A machine learning model is defined as a mathematical representation of the output of the training process. Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

Machine Learning models can be understood as a program that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response. First, these models are trained over a set of data, and then they are provided an algorithm to reason over data, extract the pattern from feed data and learn from those data. Once these models get trained, they can be used to predict the unseen dataset.

There are various types of machine learning models available based on different business goals and data sets.

```
In [7]: X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)

(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

```
In [8]: 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))
model.add(Dense(43, activation='softmax'))
model.summary()

Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	2432
conv2d_1 (Conv2D)	(None, 22, 22, 32)	25632

Figure 20 Loading the model

## Training the model

Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization.

Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples.



In Machine Learning, whenever you want to train a model with some data, then Epoch refers to one complete pass of the training dataset through the algorithm. Moreover, it takes a few epochs while training a machine learning model, but, in this scenario, you will face an issue while feeding a bunch of training data in the model. This issue happens due to limitations of computer storage. To overcome this issue, we have to break the training data into small batches according to the computer memory or storage capacity. Then only we can train a machine learning model by feeding these batches without any hassle. This process is called batch in machine learning, and further, when all batches are fed exactly once to train the model, then this entire procedure is known as Epoch in Machine Learning. Epochs are defined as the total number of iterations for training the machine learning model with all the training data in one cycle. In Epoch, all training data is used exactly once. Further, in other words, Epoch can also be understood as the total number of passes an algorithm has completed around the training dataset. A forward and a backward pass together counted as one pass in training.

Usually, when a machine learning model is trained, then it requires a little number of Epochs. An Epoch is often mixed up with iteration.

What is Iteration?

Iteration is defined as a total number of batches required to complete one epoch, where a number of batches are equal to the total number of iterations for one epoch.

Let's understand the iteration and epoch with an example, where we have 3000 training examples that we are going to use to train a machine learning model.

In the above scenario, we can break up the training dataset into sizeable batches. So let's suppose we have considered the batches of 500 examples in each batch, then it will take 6 iterations to complete 1 Epoch.

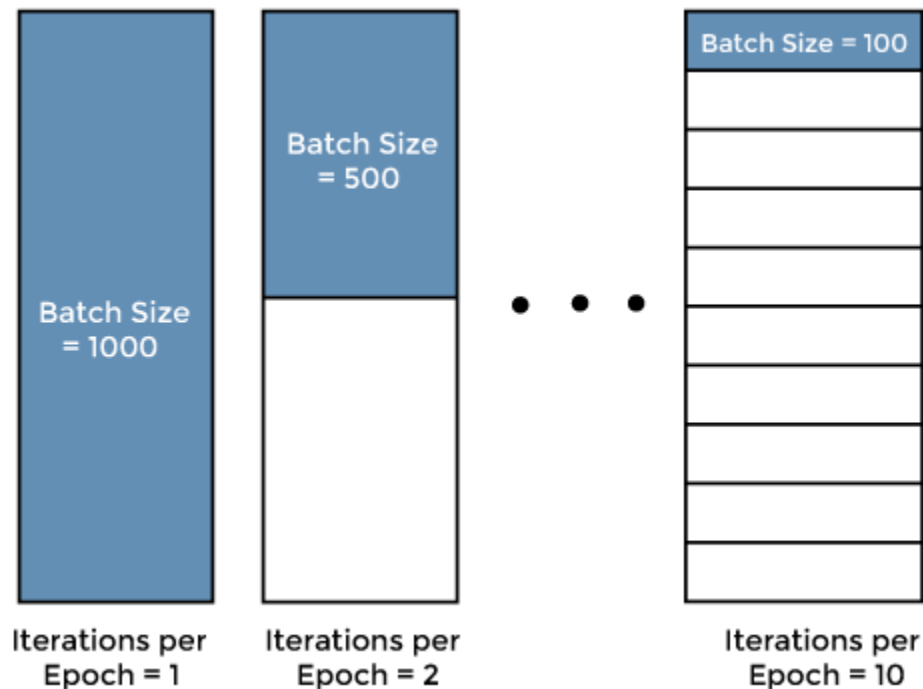
Mathematically, we can understand it as follows:

- Total number of training examples = 3000;
- Assume each batch size = 500;
- Then the total number of Iterations = Total number of training examples/Individual batch size =  $3000/500$
- Total number of iterations = 6
- And 1 Epoch = 6 Iterations

What is Batch in Machine Learning?

Batch size is defined as the total number of training examples that exist in a single batch. You can understand batch with the above-mentioned example also, where we have divided the entire training dataset/examples into different batches or sets or parts.

Let's understand the concept of mixing up an Epoch and iteration with the below example where we have considered 1000 datasets as shown in the below image.



*Figure 21 Batch example*

In the above figure, we can understand this concept as follows:

- If the Batch size is 1000, then an epoch will complete in one iteration.
- If the Batch size is 500, then an epoch will complete in 2 iterations.

Similarly, if the batch size is too small or such as 100, then the epoch will be complete in 10 iterations. So, as a result, we can conclude that for each epoch, the required number of iterations times the batch size gives the number of data points. However, we can use multiple numbers epochs for training the machine learning model.

There are a few important points that everyone should keep in mind during training a machine learning model. These are as follows:

- Epoch is a machine learning terminology that refers to the number of passes the training data goes through machine learning algorithm during the entire data points.
- If there is a large amount of data available, then you can divide entire data sets into common groups or batches.
- The process of running one batch through the learning model is known as iteration. In Machine Learning, one cycle in entire training data sets is called an Epoch. However, in ideal conditions, one cycle in entire training data sets is called an Epoch but training a model typically requires multiple numbers of Epochs.
- Better generalization can be achieved with new inputs by using more epochs in the training of the machine learning model.
- Given the complexity and variety of data in real-world applications, hundreds to thousands of epochs may be required to achieve reasonable test data correctness. Furthermore, the term epoch has several definitions depending on the topic at hand.

Why use more than one Epoch?

It may not look correct that passing the entire dataset through an ML algorithm or neural network is not enough, and we need to pass it multiple times to the same algorithm.

So it needs to be kept in mind that to optimize the learning, we use gradient descent, an iterative process. Hence, it is not enough to update the weights with a single pass or one epoch.

Moreover, one epoch may lead to overfitting in the model.

```
In [9]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

In [10]: epochs = 15
         history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
         model.save("my_model.h5")

Epoch 1/15
981/981 [=====] - 39s 39ms/step - loss: 2.3195 - accuracy: 0.3857 - val_loss: 1.1013 - val_accuracy: 0.7105
Epoch 2/15
981/981 [=====] - 39s 40ms/step - loss: 1.2025 - accuracy: 0.6380 - val_loss: 0.5953 - val_accuracy: 0.8527
Epoch 3/15
981/981 [=====] - 40s 41ms/step - loss: 0.8049 - accuracy: 0.7503 - val_loss: 0.3111 - val_accuracy: 0.9149
Epoch 4/15
981/981 [=====] - 40s 41ms/step - loss: 0.5940 - accuracy: 0.8183 - val_loss: 0.1960 - val_accuracy: 0.9452
Epoch 5/15
981/981 [=====] - 39s 40ms/step - loss: 0.4912 - accuracy: 0.8500 - val_loss: 0.1727 - val_accuracy: 0.9527
Epoch 6/15
981/981 [=====] - 40s 40ms/step - loss: 0.4176 - accuracy: 0.8728 - val_loss: 0.1354 - val_accuracy: 0.9616
Epoch 7/15
981/981 [=====] - 40s 40ms/step - loss: 0.3758 - accuracy: 0.8887 - val_loss: 0.1768 - val_accuracy: 0.9500
Epoch 8/15
981/981 [=====] - 40s 40ms/step - loss: 0.3209 - accuracy: 0.9055 - val_loss: 0.0989 - val_accuracy:
```

Figure 22 Training the model

## Results and Reports

The model has been trained, now, we will make a separate file main.py which will load the model create the GUI.

```
1 import tkinter as tk
2 from tkinter import filedialog
3 from tkinter import *
4 from PIL import ImageTk, Image
5 import numpy as np
6
7 from keras.models import load_model
8 model = load_model(r"")
9
10
11 classes = { 1:'Speed limit (20km/h)',
12            2:'Speed limit (30km/h)',
13            3:'Speed limit (50km/h)',
14            4:'Speed limit (60km/h)',
15            5:'Speed limit (70km/h)',
16            6:'Speed limit (80km/h)',
17            7:'End of speed limit (80km/h)',
18            8:'Speed limit (100km/h)',
19            9:'Speed limit (120km/h)',
20            10:'No passing',
21            11:'No passing veh over 3.5 tons',
22            12:'Right-of-way at intersection',
23            13:'Priority road',
24            14:'Yield',
25            15:'Stop',
26            16:'No vehicles',
27            17:'Veh > 3.5 tons prohibited',
28            18:'No entry',
29            19:'General caution',
30            20:'Dangerous curve left',
31            21:'Dangerous curve right',
32            22:'Double curve',
33            23:'Bumpy road',
34            24:'Slippery road',
35            25:'Road narrows on the right',
36            26:'Road work',
```

Figure 23 building the GUI

```
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')

label=Label(top,background='#CCCC', font=('arial',15,'bold'))
sign_image = Label(top)

def classify(file_path):
    global label_packed
    image = image.open(file_path)
    image = image.resize((30,30))

    image = np.expand_dims(image, axis=0)
    image = np.array(image)
    print(image.shape)
    print(image.shape[3])
    if image.shape[3] == 4:
        image = image[:, :, :, :3]
        print(image.shape)

    pred=model.predict(image)
    print(pred)
    x = np.argmax(pred)
    print(x)
    sign = classes[x+1]

    print(sign)
    label.configure(foreground='#011638', text=sign)

def show_classify_button(file_path):
    Classify_b=Button(top,text="Classify Image",command=lambda: classify(file_path),padx=10,pady=5)
    Classify_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
```

Figure 24 building the GUI

## Snapshots

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
```

Figure 25 Importing the libraries

```
In [10]: epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
model.save("my_model.h5")

Epoch 1/15
981/981 [=====] - 39s 39ms/step - loss: 2.3195 - accuracy: 0.3857 - val_loss: 1.1013 - val_accuracy:
0.7105
Epoch 2/15
981/981 [=====] - 39s 40ms/step - loss: 1.2025 - accuracy: 0.6380 - val_loss: 0.5953 - val_accuracy:
0.8527
Epoch 3/15
981/981 [=====] - 40s 41ms/step - loss: 0.8049 - accuracy: 0.7503 - val_loss: 0.3111 - val_accuracy:
0.9149
Epoch 4/15
981/981 [=====] - 40s 41ms/step - loss: 0.5940 - accuracy: 0.8183 - val_loss: 0.1960 - val_accuracy:
0.9452
Epoch 5/15
981/981 [=====] - 39s 40ms/step - loss: 0.4912 - accuracy: 0.8500 - val_loss: 0.1727 - val_accuracy:
0.9527
Epoch 6/15
981/981 [=====] - 40s 40ms/step - loss: 0.4176 - accuracy: 0.8728 - val_loss: 0.1354 - val_accuracy:
0.9616
Epoch 7/15
981/981 [=====] - 40s 40ms/step - loss: 0.3758 - accuracy: 0.8887 - val_loss: 0.1768 - val_accuracy:
0.9500
Epoch 8/15
981/981 [=====] - 40s 40ms/step - loss: 0.3209 - accuracy: 0.9055 - val_loss: 0.0989 - val_accuracy:
0.9744
```

Figure 26 Training steps



Figure 27 TSR GUI



Figure 28 Uploading an image to GUI



Figure 29 Classify the image

## Testing and Verification

Accuracy is perhaps the best-known Machine Learning model validation method used in evaluating classification problems. One reason for its popularity is its relative simplicity. It is easy to understand and easy to implement. Accuracy is a good metric to assess model performance in simple cases.

Accuracy is used in classification problems to tell the percentage of correct predictions made by a model. Accuracy score in machine learning is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made. We calculate it by dividing the number of correct predictions by the total number of predictions.

It is not perfect, but does offer 5 compelling benefits:

- ❖ It is quick and inexpensive to calculate.
- ❖ It is easy to understand.
- ❖ It is language independent.
- ❖ It correlates highly with human evaluation.
- ❖ It has been widely adopted.

In [11]:

```
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```

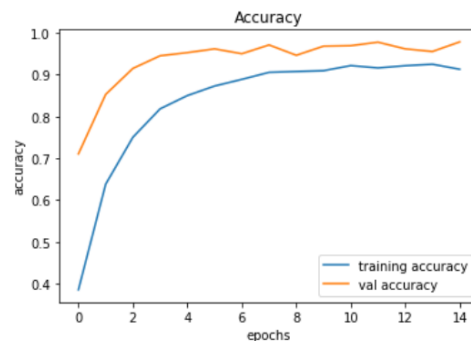


Figure 30 Accuracy diagram of the model

```
In [12]: plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

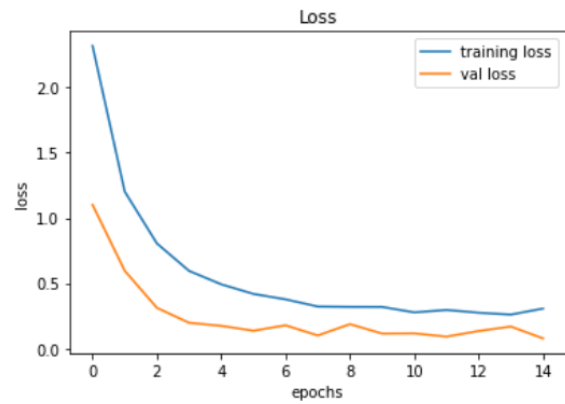


Figure 31 Data loss diagram



## **CONCLUSION**

After building the model and implementing the project we got the expected results, as we got a very high accuracy in detecting the Traffic signals. And after testing the GUI we found it very reliable and user friendly

## **Scope of future work**

Recently, there is redundant research work emerging for recognizing traffic signs for handling with the real-world problems. On the one hand, in future, we will complete our benchmark by covering more types of the traffic signs so that we can make this project more instructional in this field. On the other hand, more object recognition techniques will be employed into TSR. For example, recognizing objects utilizes heatmaps methods. Finally, more evaluation measures also should be used to estimate the performance of different models.

## REFERENCES

- [1] Zhu, Y., Zhang, C., Zhou, D., Wang, X., Bai, X. and Liu, W., 2016. Traffic sign detection and recognition using fully convolutional network guided proposals. *Neurocomputing*, 214, pp.758-766.
- [2] M. M. Lau, K. H. Lim and A. A. Gopalai, "Malaysia traffic sign recognition with convolutional neural network," 2015 IEEE International Conference on Digital Signal Processing (DSP), 2015, pp. 1006-1010, doi: 10.1109/ICDSP.2015.7252029.
- [3] Vennelakanti, Aashrith, et al. "Traffic sign detection and recognition using a CNNS ensemble." 2019 IEEE international conference on consumer electronics (ICCE). IEEE, 2019.
- [4] Luo, Hengliang, et al. "Traffic sign recognition using a multi-task convolutional neural network." *IEEE Transactions on Intelligent Transportation Systems* 19.4 (2017): 1100-1111.
- [5] Qian, Rongqiang, et al. "Robust Chinese traffic sign detection and recognition with deep convolutional neural network." 2015 11th International Conference on Natural Computation (ICNC). IEEE, 2015.
- [6] Alghmgham, Danyah A., et al. "Autonomous traffic sign (ATSR) detection and recognition using deep CNNS." *Procedia Computer Science* 163 (2019): 266-274.
- [7] Jin, Junqi, Kun Fu, and Changshui Zhang. "Traffic sign recognition with hinge loss trained convolutional neural networks." *IEEE transactions on intelligent transportation systems* 15.5 (2014): 1991-2000.
- [8] Qian, Rongqiang, et al. "Traffic sign recognition with convolutional neural network based on max pooling positions." 2016 12th International conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD). IEEE, 2016.
- [9] Shustanov, Alexander, and Pavel Yakimov. "CNNS design for real-time traffic sign recognition." *Procedia engineering* 201 (2017): 718-725.
- [10] Wu, Yihui, et al. "Traffic sign detection based on convolutional neural networks." *The 2013 international joint conference on neural networks (IJCNN)*. IEEE, 2013.
- [11] <https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>