**Road Safety (sign recognition)**

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

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

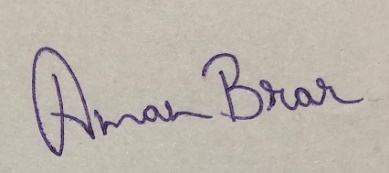


**Submitted to:**

**Amandeep Kaur**

**Mentor:**

**Amandeep Kaur**

****

**Submitted By:**

**Aditya Arora(18BCS6656)**

**Shivam badoniya (18BCS3054)**

**kanishka singh rathore (18BCS6634)**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Chandigarh University, Gharuan**

**April 2021**

**Project Design**

You must have heard about the self-driving cars in which the passenger can fully depend on the car for traveling. But to achieve level 5 autonomous, it is necessary for vehicles to understand and follow all traffic rules.

In the world of Artificial Intelligence and advancement in technologies, many researchers and big companies like Tesla, Uber, Google, Mercedes-Benz, Toyota, Ford, Audi, etc are working on autonomous vehicles and self-driving cars. So, for achieving accuracy in this technology, the vehicles should be able to interpret traffic signs and make decisions accordingly.

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 example, 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.

Traditionally, standard [computer vision](https://en.wikipedia.org/wiki/Computer_vision) methods were employed to detect and classify traffic signs, but these required considerable and time-consuming manual work to handcraft important features in images. Instead, by applying deep learning to this problem, we create a model that reliably classifies traffic signs, learning to identify the most appropriate features for this problem by *itself*. In this post, I show how we can create a deep learning architecture that can identify traffic signs with close to 98% accuracy on the test set.



**Constraints:**

The application was built on and for an Window.

The system also assumes that the user has minimal English knowledge as of now.

### The Dataset of Python Project:

For this project, we are using the public dataset available at Kaggle:

[Traffic Signs Dataset](https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign)

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 300 MB. The dataset has a train folder which contains images inside each class and a test folder which we will use for testing our model.

# Project Setup

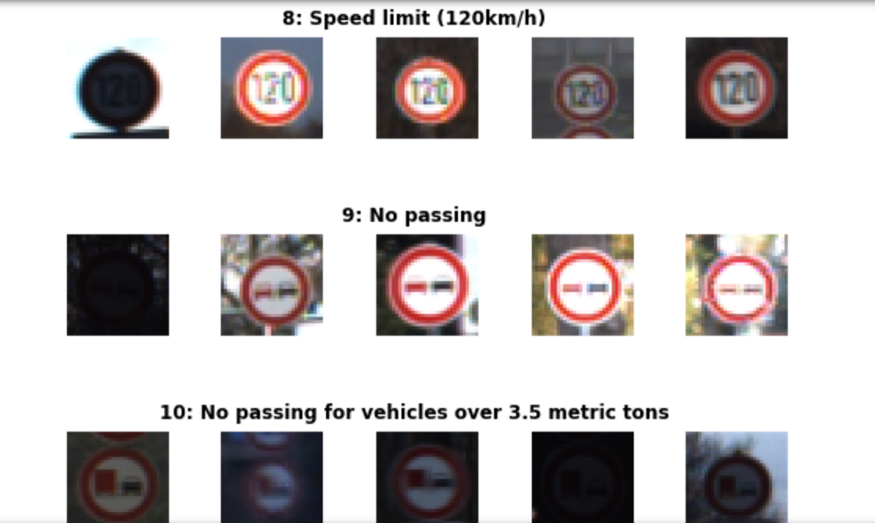
The dataset is plit into training, test and validation sets, with the following characteristics:

* Images are 32 (width) x 32 (height) x 3 (RGB color channels)
* Training set is composed of 34799 images
* Validation set is composed of 4410 images
* Test set is composed of 12630 images
* There are 43 classes (e.g. Speed Limit 20km/h, No entry, Bumpy road, etc.)

Moreover, we will be using Python 3.5 with Tensorflow to write our code.

## Images And Distribution

You can see below a sample of the images from the dataset, with labels displayed above the row of corresponding images. Some of them are quite dark so we will look to improve contrast a bit later.

****

**Conceptual framework:**

There is always scope for improvement. So the application was built using M.V.C architecture, incremental process design. These design structures follow the modular approach which makes it easy to add in new features to the system.

The system also implements the singleton pattern and the single responsibility principle which ensure the individual functioning of the modules.

Functions:

**The system provides all the below stated functions.**

* -displaying system information such as network usage, ram etc.
* -opening and closing an application
* -altering system brightness, volume
* -creating a new window or a tab in browser
* -going to the previous webpage on the web browser
* -reloading a webpage on the web browser
* -keylogging with keystrokes segregated according to the application and time stamps.
* sign recognition

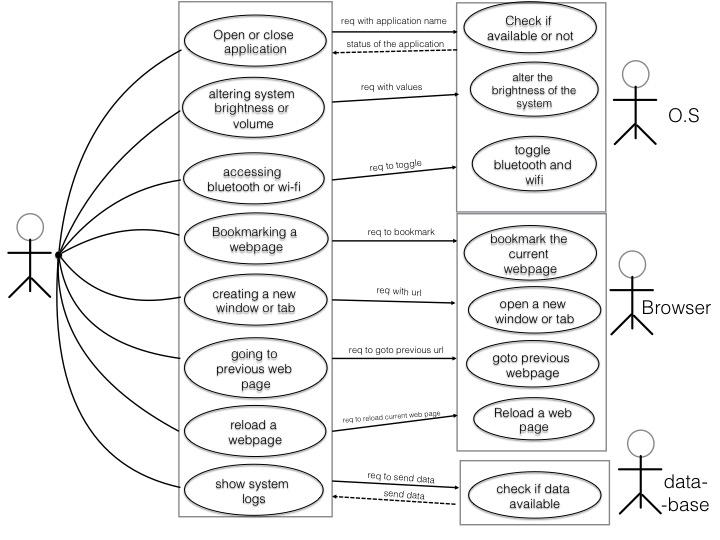
**Innovation in model/design/solution**

**Innovation in Project:**

Traffic sign recognition systems have been controversial in recent months after the European Union proposed laws which would use them to automatically limit the speed of all new cars manufactured after 2022.

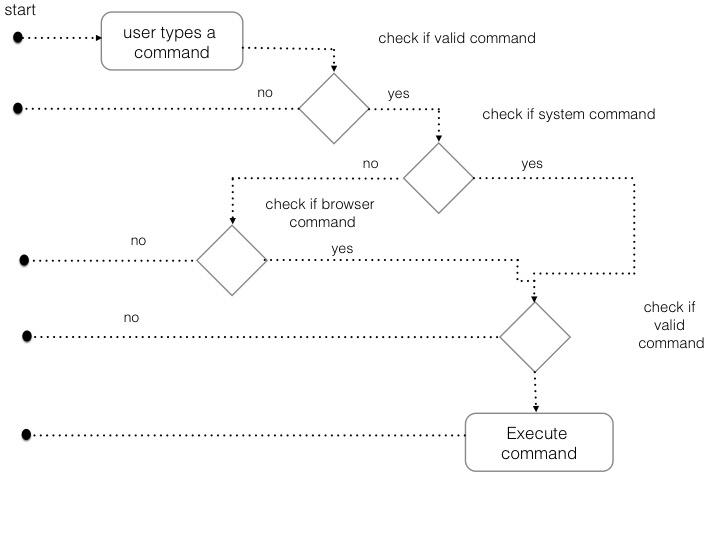
The law, which has only been provisionally agreed but which the Department for Transport have said would be implemented regardless of the outcome of Brexit, insists new cars use ‘Intelligent Speed Assistance’ (ISA), a hybrid system that uses GPS and traffic sign recognition to stop cars going over the speed limit. Whilst the system can be overridden, it’s hoped its implementation will save around 25,000 lives within 15 years.

**Diagrams: UML Diagrams:  
Use Case Diagram:**A use case is a set of scenarios that describe an interaction between a user and a system.  A use case diagram displays the relationship among actors and use cases.  The two main components of a use case diagram are use cases and actors.

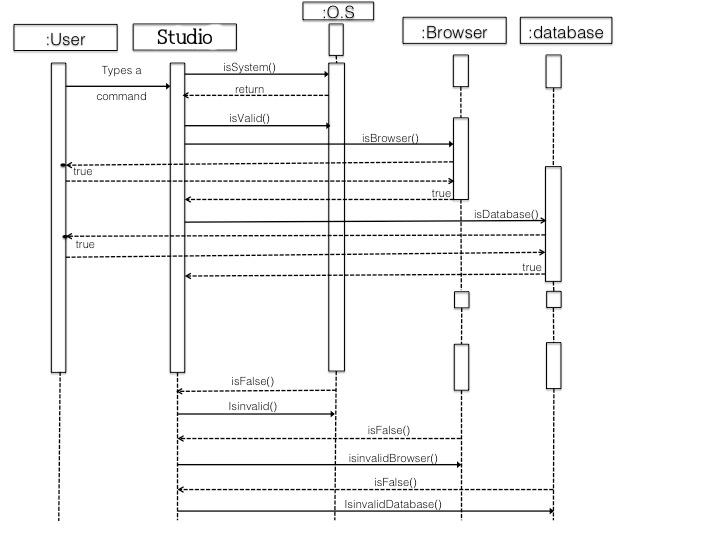


**Activity Diagram:**

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc



**Sequence diagram**A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart.  
A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario



**Implementation**

I am going to use the VS Code IDE in this video. Feel free to use any other IDE you are comfortable d with. Start a new project and make a file called Jarvis.py.

We are going to do step by step implementation also we will going to install necessary libraries

#### Defining Speak Function

The and first and foremost thing for an A.I. assistant is that it should be able to speak. To make our J.A.R.V.I.S. talk, we will make a function called **speak().**This function will take audio as an argument, and then, it will pronounce it.

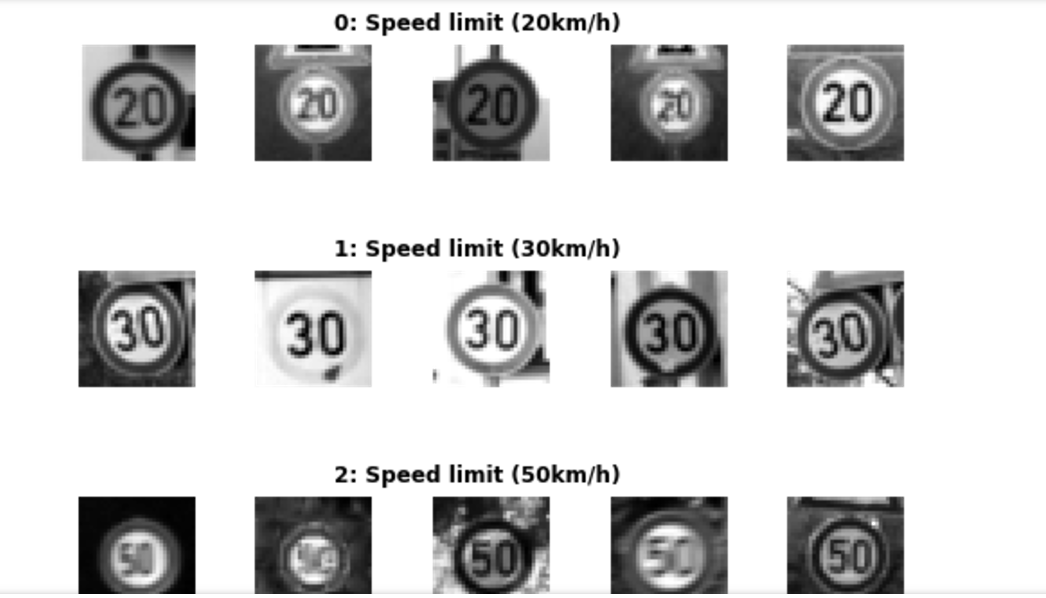
Now, the next thing we need is audio. We must supply audio so that we can pronounce it using the speak() function we made. We are going to install a module called **pyttsx3.**

# Pre-Processing Steps

We initially apply two pre-processing steps to our images:

**Grayscale**

We convert our 3 channel image to a single grayscale image (we do the same thing in project 1 — Lane Line Detection — you can read my blog post about it.



Sample Of Grayscale Training Set Images, with labels above

**Image Normalisation**

We center the distribution of the image dataset by subtracting each image by the dataset mean and divide by its standard deviation. This helps our model treating images uniformly. The resulting images look as follows.

