**Abstract**

To ensure a smooth and secure flow of traffic, road signs are essential. A major cause of road accidents is negligence in viewing the Traffic signboards and interpreting them incorrectly. The proposed system helps in recognizing the Traffic sign and sending a voice alert through the speaker to the driver so that he/ she may take necessary decisions. The proposed system is trained using Convolutional Neural Network (CNN) which helps in traffic sign image recognition and classification. A set of classes are defined and trained on a particular dataset to make it more accurate. The German Traffic Sign Benchmarks Dataset was used, which contains approximately 43 categories and 51,900 images of traffic signs. The accuracy of the execution is about 98.52 percent. Following the detection of the sign by the system, a voice alert is sent through the speaker which notifies the driver. The proposed system also contains a section where the vehicle driver is alerted about the traffic signs in the near proximity which helps them to be aware of what rules to follow on the route. The aim of this system is to ensure the safety of the vehicle's driver, passengers, and pedestrians.

**CHAPTER 1**

**INTRODUCTION**

Traffic signs detection and recognition is an important aspect for providing safety for all the road users. Numerous computer vision systems have been developed recently for traffic signs analysis. But the characteristics of existing algorithms (recognition accuracy, false alarm percentage, robustness against atmospheric changes) are still not good enough to substitute a human operator.

Over the world, 1.2 million people were killed in traffic crashes in 2002, which was 2.1% of all global deaths and the 11th ranked cause of death [1]. Nowadays, we are moving towards a new era in which, thanks to technologies, crashes are rare rather then commonplace. In fact, new Intelligent Transport Systems (ITS) have been introduced in automotive industry in order to save money and lives, and to make the driving safe and convenient.

There have been a lot of technological advancements and cars with auto-pilot mode have come up. Autonomous vehicles have come into existence. There has been a boom in the self-driving car industry. However, these features are available only in some high-end cars which are not affordable to the masses. We wanted to devise a system which helps in easing the job of driving to some extent.

On conducting a survey, we found that the magnitude of road accidents in India is alarming. Reports suggest that every hour there are about 53 mishaps taking place on the roads. Moreover, every hour more than 16 deaths occur due to these mishaps. When someone neglects to obey traffic signs while driving, they are putting their life as well the life of the other drivers, their passengers and those on the road at risk. Hence, we came up with this system in which traffic signs are automatically detected using the live video stream and are read out aloud to the driver who may then take the required decision. Another area of focus in our system is the idea of getting the location of the user using GPS. Also, all the traffic signs will be stored in a database along with their location so that the driver will be notified in advance regarding the next approaching Traffic Sign.



Fig. 1 Traffic Signs Taken into consideration

**Problem statement**

To ensure a smooth and secure flow of traffic, road signs are essential. A major cause of road accidents is negligence in viewing the Traffic signboards and interpreting them incorrectly. A system that can help in recognizing the Traffic sign and sending a voice alert through the speaker to the driver so that he/ she may take necessary decisions.

**Scope:**

Traffic sign detection and recognition plays an important role in expert systems, such as traffic assistance driving systems and automatic driving systems. It instantly assists drivers or automatic driving systems in detecting and recognizing traffic signs effectively.

The traffic sign board recognition focuses on reduction of the traffic load on existingroad network through various travel demand management measures.

**Application:**

1. This proposed system is a significant advancement in the field of driving as it would ease the job of the driver without compromising on the safety aspect.

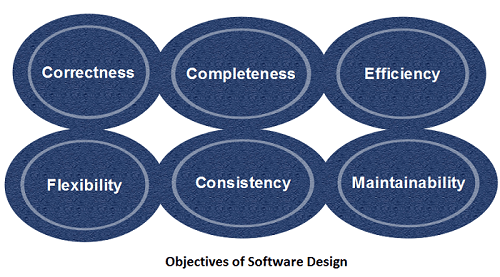
**DESIGN**

 Software design is **the process of envisioning and defining software solutions to one or more sets of problems**. One of the main components of software design is the software requirements analysis (SRA). SRA is a part of the software development process that lists specifications used in software engineering.

The software design phase is the first step in **SDLC (Software Design Life Cycle)**, which moves the concentration from the problem domain to the solution domain. In software design, we consider the system to be a set of components or modules with clearly defined behaviors & boundaries.

Objectives of Software Design

Following are the purposes of Software design:



1. **Correctness:** Software design should be correct as per requirement.
2. **Completeness:** The design should have all components like data structures, modules, and external interfaces, etc.
3. **Efficiency:** Resources should be used efficiently by the program.
4. **Flexibility:** Able to modify on changing needs.
5. **Consistency:** There should not be any inconsistency in the design.
6. **Maintainability:** The design should be so simple so that it can be easily maintainable by other designers

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

**THE DESIGN PROCESS CONSISTS OF 6 STEPS:**

* Define the Problem. You can't find a solution until you have a clear idea of what the problem is.
* Collect Information. Collect sketches, take photographs and gather data to start giving you inspiration.
* Brainstorm and Analyse Ideas.
* Develop Solutions.
* Gather Feedback.
* Improve.

A CNN architecture is **formed by a stack of distinct layers that transform the input volume into an output volume (e.g., holding the class scores) through a differentiable function**. A few distinct types of layers are commonly used.

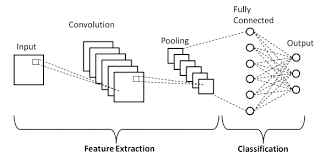


Fig: CNN Architecture

The architecture of a Convolutional Network can be compared to the connectivity pattern of Neurons in the Human Brain. The design itself was inspired by the organization of neurons as present in the Visual Cortex of the human brain. The neurons respond to stimuli only in a certain region of the field of view which is known as the Receptive Field. The visual area is a collection of a number of such receptive fields which help us in viewing objects. Once the model is trained over a series of epochs i.e., iterations, it develops the ability to distinguish between the dominating features and certain low-level features in the images. Based on this training, the model classifies them using the Softmax Classification technique.

Below Figure represents the number of layers used in the model. There are 4 convolution layers and 2 max pooling layers along with dropout, flatten and dense layers. Adam optimizer is used in the neural network. The input size of the image is 30\*30\*1. The model employs the ReLU activation function. We obtain a fully connected layer after the Flatten layer. and finally, the output is determined by using the softmax activation function. D. Proposed Solution Fig. 4 demonstrates the accuracy of the trained network. This model turned out to give the best accuracy as compared to the other models that we analyzed. Our model build gives an Accuracy of the on running for 20 epochs.

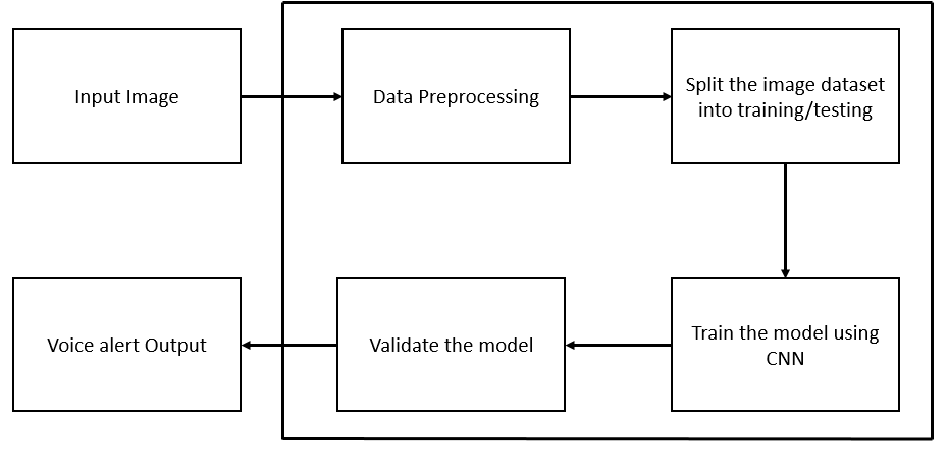


Fig. 3. Neural Network Representation

The over all design of our project is explained using the below diagrams.

The CNN model is applied in the first part wherein the input is an image. After being processed, one of the classes out of the 43 classes is obtained as the output. If a certain image is not containing a traffic sign, then the user gets a prompt of “No Sign Detected”. This is done by analyzing the After multiple iterations, it was found that even if an image does not fall in any of the given classes, the model, not being trained for an extra other class, classifies it into one of the 43 classes, but the value predicted by the “model. Predict” function is pretty low. So, the threshold value for separating the images which actually do not have a traffic sign in them but are predicted as one is taken as 0.68. The value of the classes in "model. Predict" is in the range 0 to 1 and hence if the model classifies it in a particular class with a value less than 0.68 it will be identified as none of the above, else it will be assigned a class. Once the image has been classified, the meta data is fetched from the image using “exif-parser” and then the sign text along with the GPS coordinates are stored in the database. All this data is then available to the user in a map. The map has markers containing the latitude and longitude along with the name of the traffic sign. Another important feature that needs to be highlighted here is that the aim of the proposed system is to alert the drivers. Therefore, rather than just alerting about the sign which the car is approaching i.e. the sign which has been detected, an algorithm in which the traffic signs that are in proximity i.e. the ones that will be approached within the next 5 minutes (or 1 km) are also to be alerted to the driver, is implemented. The computation of this is done by taking into account the locations of the signs which was stored by extracting the metadata.

**System Architecture**



**Fig: System Architecture**

Below are our Project Design Modules:

1. **Data Preprocessing**

Here are seven significant steps in data pre-processing in Machine Learning:

1. Acquire the dataset
2. Import all the crucial libraries.
3. Import the dataset.
4. Identifying and handling the missing values.
5. Encoding the categorical data.
6. Splitting the dataset
7. Feature scaling.

Below are data pre-processing steps to be taken for Image Dataset:

1. Read image.
2. Resize image.
3. Remove noise (Denoise)
4. Segmentation.
5. Morphology (smoothing edges)
6. **Splitting Dataset into Training & testing Dataset**

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modelling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

* Train Dataset: Used to fit the machine learning model.
* Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

The train-test procedure is appropriate when there is a sufficiently large dataset available.

The idea of “sufficiently large” is specific to each predictive modelling problem. It means that there is enough data to split the dataset into train and test datasets and each of the train and test datasets are suitable representations of the problem domain. This requires that the original dataset is also a suitable representation of the problem domain.

A suitable representation of the problem domain means that there are enough records to cover all common cases and most uncommon cases in the domain. This might mean combinations of input variables observed in practice. It might require thousands, hundreds of thousands, or millions of examples.

Conversely, the train-test procedure is not appropriate when the dataset available is small. The reason is that when the dataset is split into train and test sets, there will not be enough data in the training dataset for the model to learn an effective mapping of inputs to outputs. There will also not be enough data in the test set to effectively evaluate the model performance. The estimated performance could be overly optimistic (good) or overly pessimistic (bad).

If you have insufficient data, then a suitable alternate model evaluation procedure would be the k-fold cross-validation procedure.

In addition to dataset size, another reason to use the train-test split evaluation procedure is computational efficiency.

Some models are very costly to train, and in that case, repeated evaluation used in other procedures is intractable. An example might be deep neural network models. In this case, the train-test procedure is commonly used.

Alternately, a project may have an efficient model and a vast dataset, although may require an estimate of model performance quickly. Again, the train-test split procedure is approached in this situation.

Samples from the original training dataset are split into the two subsets using random selection. This is to ensure that the train and test datasets are representative of the original dataset.

How to Configure the Train-Test Split

The procedure has one main configuration parameter, which is the size of the train and test sets. This is most commonly expressed as a percentage between 0 and 1 for either the train or test datasets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

There is no optimal split percentage.

You must choose a split percentage that meets your project’s objectives with considerations that include:

* Computational cost in training the model.
* Computational cost in evaluating the model.
* Training set representativeness.
* Test set representativeness.

Nevertheless, common split percentages include:

* Train: 80%, Test: 20%
* Train: 67%, Test: 33%
* Train: 50%, Test: 50%

Now that we are familiar with the train-test split model evaluation procedure, let’s look at how we can use in Python in our project.

In the proposed system, the German Traffic Sign Benchmarks (GTSRB) Dataset is used. With 43 different traffic signs that are considered to train the model. It has 51,900 single images distributed among the 43 classes including the training and the test dataset. There is no ambiguity as the images are just focused on the traffic signs and each of them is unique. The training dataset has different folders for each of the present classes. A CSV file is also present wherein the path of each image and its class and other details such as width, height is mentioned.

1. **CNN Model Building/Training the Model**

These are the steps used to training the CNN (Convolutional Neural Network).

Step 1: Upload Traffic sign image Dataset.

Step 2: The Input layer.

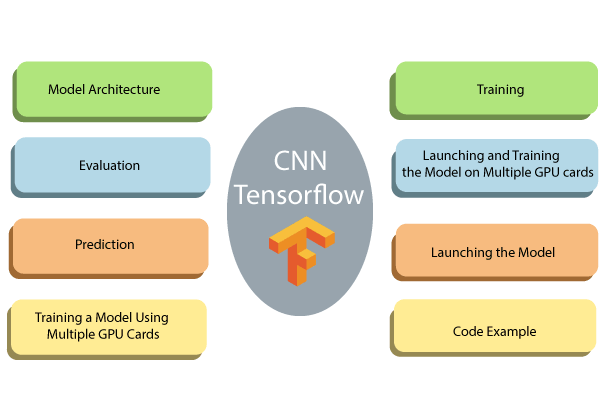
Step 3: Convolutional layer.

Step 4: Pooling layer.

Step 5: Convolutional layer and Pooling Layer.

Step 6: Dense layer.

Step 7: Logit Layer.



Defining the CNN (Convolutional Neural Network)

CNN uses filters on the pixels of any image to learn detailed patterns compared to global patterns with a traditional neural network. To create CNN, we have to define:

1. **A convolutional Layer:** Apply the number of filters to the feature map. After convolution, we need to use a relay activation function to add non-linearity to the network.
2. **Pooling Layer:** The next step after the Convention is to down sampling the maximum facility. The objective is to reduce the mobility of the feature map to prevent overfitting and improve the computation speed. Max pooling is a traditional technique, which splits feature maps into subfields and only holds maximum values.
3. **Fully connected Layers:** All neurons from the past layers are associated with the other next layers. The CNN has classified the label according to the features from convolutional layers and reduced with any pooling layer.

CNN Architecture

* **Convolutional Layer:** It applies 14 5x5 filters (extracting 5x5-pixel sub-regions),
* **Pooling Layer:** This will perform max pooling with a 2x2 filter and stride of 2 (which specifies that pooled regions do not overlap).
* **Convolutional Layer:** It applies 36 5x5 filters, with ReLU activation function
* **Pooling Layer:** Again, performs max Pooling with a 2x2 filter and stride of 2.
* **1,764 neurons,** with the dropout regularization rate of 0.4 (where the probability of 0.4 that any given element will be dropped in training)
* **Dense Layer (Logits Layer):** There are ten neurons, one for each digit target class (0-9).

**Important modules to use in creating a CNN:**

1. Conv2d (). Construct a two-dimensional convolutional layer with the number of filters, filter kernel size, padding, and activation function like arguments.
2. max\_pooling2d (). Construct a two-dimensional pooling layer using the max-pooling algorithm.
3. Dense (). Construct a dense layer with the hidden layers and units

We can define a function to build CNN.

Let's see in detail how to construct every building block before wrapping everything in the function.

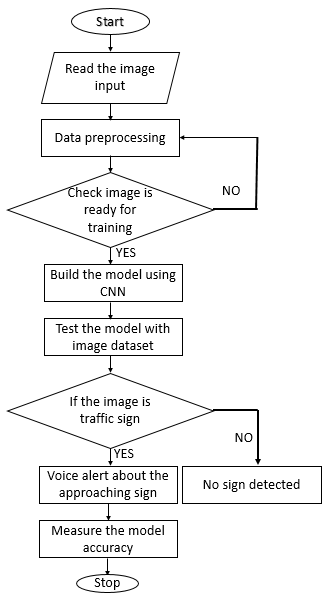
1. **Use the Model for Prediction/Model Validation**

The CNN model is applied in the first part wherein the input is an image. After being processed, one of the classes out of the 43 classes is obtained as the output. If a certain image is not containing a traffic sign, then the user gets a prompt of “No Sign Detected”. This is done by analyzing the After multiple iterations, it was found that even if an image does not fall in any of the given classes, the model, not being trained for an extra other class, classifies it into one of the 43 classes, but the value predicted by the “model. Predict” function is pretty low. So, the threshold value for separating the images which actually do not have a traffic sign in them but are predicted as one is taken as 0.68.

The value of the classes in "model. Predict" is in the range 0 to 1 and hence if the model classifies it in a particular class with a value less than 0.68 it will be identified as none of the above, else it will be assigned a class. Once the image has been classified, the meta data is fetched from the image using “exif-parser” and then the sign text along with the GPS coordinates are stored in the database.

**Data Flow Diagram**

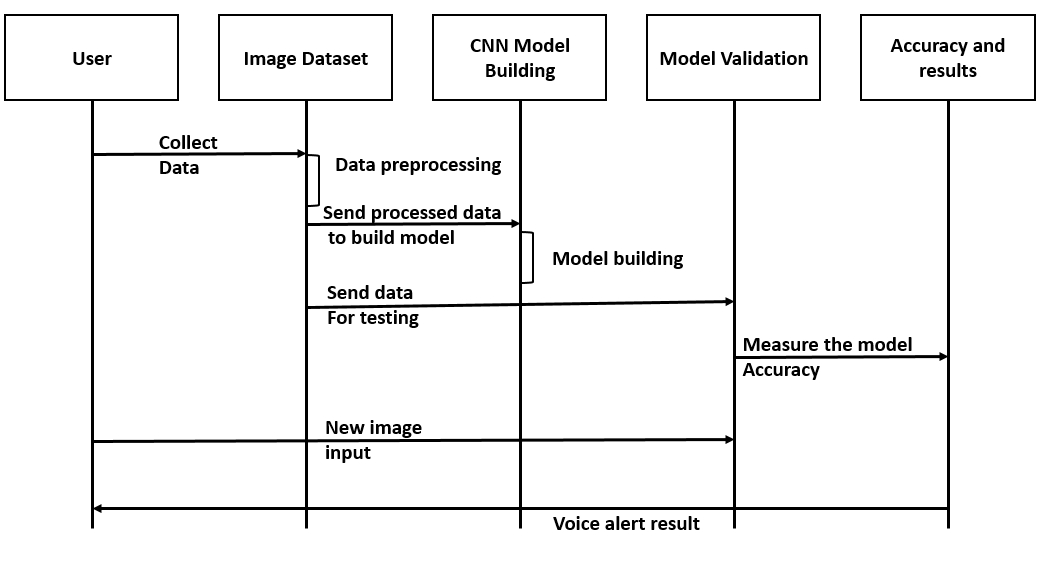
A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.



**Fig: Data Flow Diagram**

**Sequence Diagram**

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.



**Fig: Sequence Diagram**

**Usecase Diagram**

Use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. The "actors" are people or entities operating under defined roles within the system. Use case diagram with heart disease prediction system allows patient and admin.

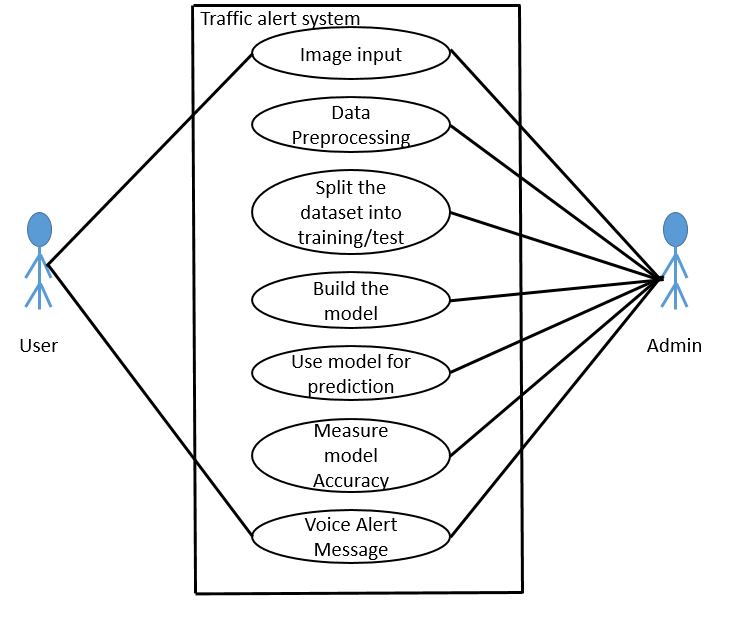


Fig: Usecase Diagram

**IMPLEMENTATION**

To ensure a smooth and secure flow of traffic, road signs are essential. A major cause of road accidents is negligence in viewing the Traffic signboards and interpreting them incorrectly. A system that can help in recognizing the Traffic sign and sending a voice alert through the speaker to the driver so that he/ she may take necessary decisions.

Traffic sign detection and recognition plays an important role in expert systems, such as traffic assistance driving systems and automatic driving systems. It instantly assists drivers or automatic driving systems in detecting and recognizing traffic signs effectively.

The traffic sign board recognition focuses on reduction of the traffic load on existingroad network through various travel demand management measures.

Entire Project is Divided into Below Modules:

* 1. UI Module
  2. Gradio
  3. Manage Module
  4. TrafficAlert Modul

**TESTING**

**REFERENCES**

* Google.com
* Wikipedia

[1] Yadav, Shubham & Patwa, Anuj & Rane, Saiprasad & Narvekar, Chhaya. (2019). Indian Traffic Sign Board Recognition and Driver Alert System Using Machine Learning. International Journal of Applied Sciences and Smart Technologies. 1. 1-10. 10.24071/ijasst.v1i1.1843.

[2] Anushree.A., S., Kumar, H., Iram, I., & Divyam, K. (2019). Automatic Signboard Detection System by the Vehicles.

[3] S. Harini, V. Abhiram, R. Hegde, B. D. D. Samarth, S. A. Shreyas and K. H. Gowranga, "A smart driver alert system for vehicle traffic using image detection and recognition technique," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2017, pp. 1540-1543, doi: 10.1109/RTEICT.2017.8256856.

[4] C. Wang, "Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning," 2018 International Conference on Robots & Intelligent System (ICRIS), Changsha, China, 2018, pp. 150-152, doi: 10.1109/ICRIS.2018.00047.

[5] M A Muchtar et al 2017 J. Phys.: Conf. Ser. 801 012010

[6] Y. Yuan, Z. Xiong and Q. Wang, "VSSA-NET: Vertical Spatial Sequence Attention Network for Traffic Sign Detection," in IEEE Transactions on Image Processing, vol. 28, no. 7, pp. 3423-3434, July 2019, doi: 10.1109/TIP.2019.2896952.

[7] S. Huang, H. Lin and C. Chang, "An in-car camera system for traffic sign detection and recognition," 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), Otsu, Japan, 2017, pp. 1-6, doi: 10.1109/IFSA-SCIS.2017.8023239.

[8] Bi, Z., Yu, L., Gao, H. et al. Improved VGG model-based efficient traffic sign recognition for safe driving in 5G scenarios. Int. J. Mach. Learn. & Cyber. (2020).

[9] Chuanwei Zhang et al., Study on Traffic Sign Recognition by Optimized Lenet-5 Algorithm, International Journal of Pattern Recognition and Artificial Intelligence, doi:0.1142/S0218001420550034

[10] Han, C., Gao, G. & Zhang, Y. Real-time small traffic sign detection with revised faster-RCNN. Multimed Tools Appl 78, 13263–13278 (2019). <https://doi.org/10.1007/s11042-018-6428-0>

[11] H. S. Lee and K. Kim, "Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 5, pp. 1652-1663, May 2018, doi: 10.1109/TITS.2018.2801560.

[12] R. Qian, Y. Yue, F. Coenen and B. Zhang, "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), Changsha, China, 2016, pp. 578-582, doi: 10.1109/FSKD.2016.7603237.

[13] A. Pon, O. Adrienko, A. Harakeh and S. L. Waslander, "A Hierarchical Deep Architecture and Mini-batch Selection Method for Joint Traffic Sign and Light Detection," 2018 15th Conference on Computer and Robot Vision (CRV), Toronto, ON, Canada, 2018, pp. 102-109, doi: 10.1109/CRV.2018.00024.

[14] Saha S., Islam M.S., Khaled M.A.B., Tairin S. (2019) An Efficient Traffic Sign Recognition Approach Using a Novel Deep Neural Network Selection Architecture. In: Abraham A., Dutta P., Mandal J., Bhattacharya A., Dutta S. (eds) Emerging Technologies in Data Mining and Information Security. Advances in Intelligent Systems and Computing, vol 814. Springer, Singapore. <https://doi.org/10.1007/978-981-13-1501-5_74>

[15] A. Welzel, A. Auerswald and G. Wanielik, "Accurate camera-based traffic sign localization," 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 2014, pp. 445-450, doi: 10.1109/ITSC.2014.6957730.

[16] M. Karaduman and H. Eren, "Deep learning based traffic direction sign detection and determining driving style," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 2017, pp. 1046-1050, doi: 10.1109/UBMK.2017.8093453.

[17] E. Winarno, W. Hadikurniawati and R. N. Rosso, "Location based service for presence system using haversine method," 2017 International Conference on Innovative and Creative Information Technology (ICITech), Salatiga, Indonesia, 2017, pp. 1-4, doi: 10.1109/INNOCIT.2017.8319153.

[18] Pal R, Ghosh A, Kumar R, et al. Public health crisis of road traffic accidents in India: Risk factor assessment and recommendations on prevention on the behalf of the Academy of Family Physicians of India. J Family Med Prim Care. 2019;8(3):775-783. doi:10.4103/jfmpc.jfmpc\_214\_18