

Traffic Sign Recognition and Voice Alert System Using CNN for Intelligent Assistance

Mr. Pathipati Chandrasekhar¹, MD.Saif², M.Krishna priya³, K.Sai varshith⁴

¹Assistant professor Computer Science Engineering & Guru Nanak Institute of Technology

²Student Computer Science Engineering & Guru Nanak Institute of Technology

³Student Computer Science Engineering & Guru Nanak Institute of Technology

⁴Student Computer Science Engineering & Guru Nanak Institute of Technology

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.

Keywords: CNN,GTRB,ADAS

1.INTRODUCTION

Traffic signs play a crucial role in regulating road safety, guiding drivers, and ensuring the smooth flow of vehicles. However, one of the leading causes of road accidents is the failure to notice or correctly interpret traffic signs, especially in high-speed or low-visibility conditions. With advancements in artificial intelligence and deep learning, automated traffic sign recognition has emerged as an effective solution to assist drivers in identifying and responding to road signs accurately. This project proposes a deep learning-based traffic sign recognition system that utilizes Convolutional Neural Networks (CNNs) to detect and classify traffic signs in real-time. The system is designed to capture and analyze

traffic sign images from live camera feeds, process them using a pre-trained CNN model, and provide instant voice alerts to drivers. By leveraging the German Traffic Sign Benchmark Dataset, which includes 43 different categories and over 51,900 images, the model is trained to recognize a wide variety of traffic signs with high accuracy. The system's primary objective is to enhance driver awareness and reduce accidents by offering real-time assistance. Unlike traditional traffic sign detection methods, which rely on manual observation, this automated system ensures faster and more reliable sign recognition, even under challenging environmental conditions. The integration of voice alerts further simplifies the driving experience by minimizing distractions and allowing drivers to focus on the road. This technology can be implemented in smart vehicles, advanced driver-assistance systems (ADAS), and autonomous driving applications to improve road safety and traffic management.

SCOPE OF THE PROJECT

The scope of this project extends to the real-time detection, recognition, and classification of traffic signs, providing audio alerts to drivers for better decision-making on the road. The system is designed to assist drivers by accurately detecting signs even in low-visibility conditions, reducing their cognitive load and ensuring quick response times. The project can be integrated into modern vehicle navigation systems, driver assistance applications, and autonomous vehicles, further enhancing road safety measures. It also has potential applications in training and testing environments for new drivers, helping them learn about traffic signs efficiently. The system can be further expanded by incorporating multilingual support for voice alerts, making it accessible to a wider audience globally.

OBJECTIVE

The objective of this project is to develop a robust and efficient traffic sign recognition system using deep learning techniques, specifically Convolutional Neural Networks (CNNs). The primary goal is to accurately classify traffic signs from the German Traffic Sign Benchmark Dataset and provide real-time voice alerts to drivers, ensuring they can take appropriate actions based on the detected signs. This system aims to minimize human errors in recognizing traffic signs, ultimately reducing road accidents. Additionally, the project focuses on optimizing model performance to achieve high accuracy while ensuring quick detection and response times. The integration of voice-based notifications further enhances driver awareness and promotes safer driving practices.

EXISTING TECHNIQUE

The existing system primarily relies on Support Vector Machine (SVM) with Template Matching for traffic sign recognition. SVM is a supervised learning algorithm used for classification tasks, where it finds the optimal hyperplane that best separates different classes. In the context of traffic sign recognition, the SVM model is trained on a dataset consisting of traffic sign images categorized based on their shapes, colors, and text patterns. The training process involves extracting features such as edges, colors, and gradients, which are then used to classify traffic signs into predefined categories. Template Matching is used to compare detected signs with a set of pre-stored reference templates, ensuring accurate identification. This approach follows a structured pipeline that includes image preprocessing, segmentation, feature extraction, and classification. Initially, an input image is segmented based on color information to extract regions that might contain traffic signs. The segmented region is then classified using SVM, and the detected sign is matched with a pre-defined template to confirm its identity. However, this method struggles with real-world variations such as illumination changes, occlusions, and deformations in traffic signs, leading to lower recognition accuracy and inefficiencies in real-time applications.

EXISTING SYSTEM

The existing traffic sign recognition systems primarily rely on traditional machine learning methods such as Support Vector Machines (SVM) and template matching. Yadav et al. proposed an SVM-based approach, where

traffic sign recognition follows a structured pipeline involving color segmentation, shape classification, and recognition. The dataset used for training was divided into a 90/10 ratio, with a linear classification model employed to differentiate traffic signs. While this approach enables recognition of standard signs, it struggles with real-world variations such as lighting conditions, occlusions, and distortions. Similarly, A.Anushree. developed a traffic sign recognition system using Raspberry Pi, making the system more compact and implementable with fewer coding requirements. However, its dependence on hardware components like Raspberry Pi increases implementation costs and limits its accessibility.

Another existing method, introduced by S. Harini et al., focuses on a video-based approach where traffic signs are detected from video frames. The system preprocesses images by enhancing contrast, separating the foreground from the background, and identifying shapes like hexagonal, triangular, or circular signs. These identified shapes are then matched against pre-trained templates to recognize traffic signs. However, existing systems face challenges due to limited datasets, as deep learning methods require large, diverse training data for robust detection. Additionally, most current solutions prioritize recognition over detection, meaning they may correctly classify a sign but struggle to pinpoint its precise location in an image or video. The absence of a large-scale dataset and an efficient detection mechanism limits the effectiveness of these approaches.

EXISTING SYSTEM DISADVANTAGES

- Limited accuracy due to manual feature extraction
- Struggles with variations in lighting and weather conditions
- High computational cost for real-time applications
- Inefficient in handling occlusions and distortions
- Requires extensive preprocessing and template matching

2.METHODOLOGIES

MODULES NAME

- Dataset
- Importing the necessary libraries
- Retrieving the images

- Splitting the dataset
- Building the model
- Apply the model and plot the graphs for accuracy and loss
- Accuracy on test set
- Voice alert
- Saving the Trained Model

MODULES EXPLANATION

Dataset: In the first module, we developed the system to get the input dataset for the training and testing purpose. We have taken the dataset from German Traffic Sign Benchmark single-image classification challenge held at the International Joint Conference on Neural Networks Link -[kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign](https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign). The dataset consists of 39,209 traffic sign images and no sign

Importing the necessary libraries: We will be using Python language for this. First we will import the necessary libraries such as keras for building the main model, sklearn for splitting the training and test data, PIL for converting the images into array of numbers and other libraries such as pandas, numpy, matplotlib and tensor flow.

Retrieving the images: We will retrieve the images and their labels. Then resize the images to (30,30) as all images should have same size for recognition. Then convert the images into numpy array.

Splitting the dataset: Split the dataset into train and test. 80% train data and 20% test data.

Building the model: For building the we will use sequential model from keras library. Then we will add the layers to make convolutional neural network. In the first 2 Conv2D layers we have used 32 filters and the kernel size is (5,5).

In the MaxPool2D layer we have kept pool size (2,2) which means it will select the maximum value of every 2 x 2 area of the image. By doing this dimensions of the image will reduce by factor of 2. In dropout layer we have kept dropout rate = 0.25 that means 25% of neurons are removed randomly.

We apply these 3 layers again with some change in parameters. Then we apply flatten layer to convert 2-D

data to 1-D vector. This layer is followed by dense layer, dropout layer and dense layer again. The last dense layer outputs 43 nodes as the traffic signs are divided into 44 categories in our dataset. This layer uses the soft max activation function which gives probability value and predicts which of the 44 options has the highest probability.

Apply the model and plot the graphs for accuracy and loss: We will compile the model and apply it using fit function. The batch size will be 32. Then we will plot the graphs for accuracy and loss. We got average validation accuracy of 94.6% and average training accuracy of 96.3%.

Accuracy on test set: We got a accuracy of 96.7% on test set.

Voice alert:pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the pyttsx3.init() factory function to get a reference to a pyttsx3. Engine instance. it is a very easy to use tool which converts the entered text into speech. The pyttsx3 module supports two voices first is female and the second is male which is provided by "sapi5" for windows.

Saving the Trained Model: Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle .Make sure you have pickle installed in your environment. Next, let's import the module and dump the model into.pkl file

3. PROPOSED METHODOLOGY

3.1 System Overview

The system follows a structured pipeline:

1. Image Acquisition – traffic sign images are captured via camera or sourced from a dataset.
2. Preprocessing – Images are converted to grayscale, resized to 224×224 pixels, and normalized.
3. Feature Extraction and Classification – The CNN model learns discriminative features and outputs class probabilities.
4. Decision Fusion – Softmax probabilities across seven signs classes are fused to yield the final prediction.

3.2 Dataset

The model is trained and evaluated on a Traffic sign recognition on dataset from german traffic sign recognition benchmark

containing labeled images across several traffic sign images

categories: stop, one way, accident prone area, etc. surprise, and neutral. The dataset is split 80/20 for training and testing

3.3 Preprocessing

Preprocessing enhances data quality and model convergence:

- Grayscale Conversion – Reduces input dimensionality.
- Resizing – Standardizes spatial dimensions to 224×224.
- Normalization – Scales pixel values to [0, 1].
- Data Augmentation – Random horizontal flips, rotations, and brightness adjustments increase robustness.

3.4 Training Configuration

- Optimizer: Adam with default learning rate (0.001)
- Loss Function: Categorical cross-entropy
- Batch Size: 10
- Epochs: Increased iteratively; accuracy gains observed up to convergence.

PROPOSED TECHNIQUE USED OR ALGORITHM USED:

The proposed system employs a Convolutional Neural Network (CNN) for detecting and recognizing traffic signs more efficiently. CNN is a deep learning algorithm specifically designed for image classification tasks, making it ideal for traffic sign recognition. Unlike SVM, CNNs automatically learn hierarchical feature representations from raw images without requiring manual feature extraction. The model consists of multiple layers, including convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. The final output layer assigns a probability score to different traffic sign categories. In the proposed system, the CNN model is trained on a large dataset of traffic sign images, allowing it to recognize various traffic sign types in real-world conditions. Once a sign is detected, the system

triggers a voice alert mechanism, providing real-time feedback to drivers about the identified traffic sign. This deep learning-based approach overcomes the limitations of traditional SVM-based methods by improving accuracy, handling diverse lighting and weather conditions, and ensuring real-time performance. Furthermore, the CNN model effectively differentiates between relevant traffic signs and non-relevant objects, reducing false detections and enhancing road safety.

UML DIAGRAMS

USE CASE DIAGRAM



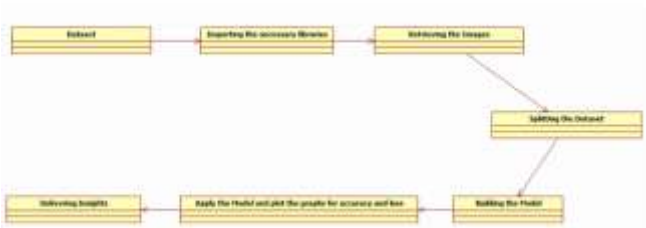
EXPLANATION: The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

CLASS DIAGRAM



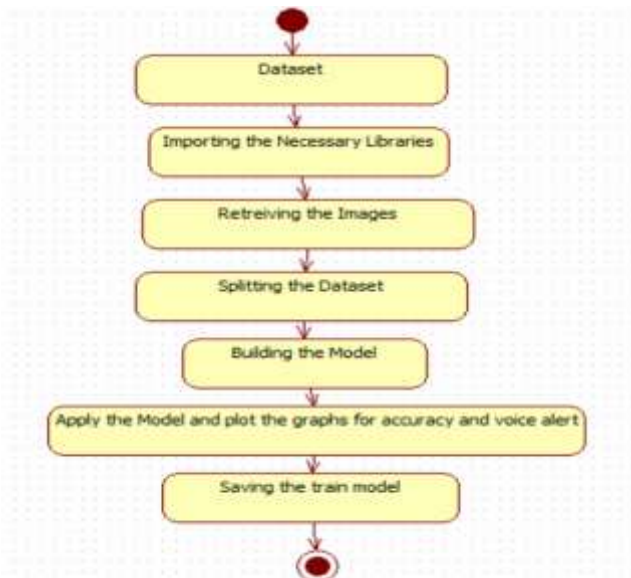
EXPLANATION: In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

OBJECT DIAGRAM



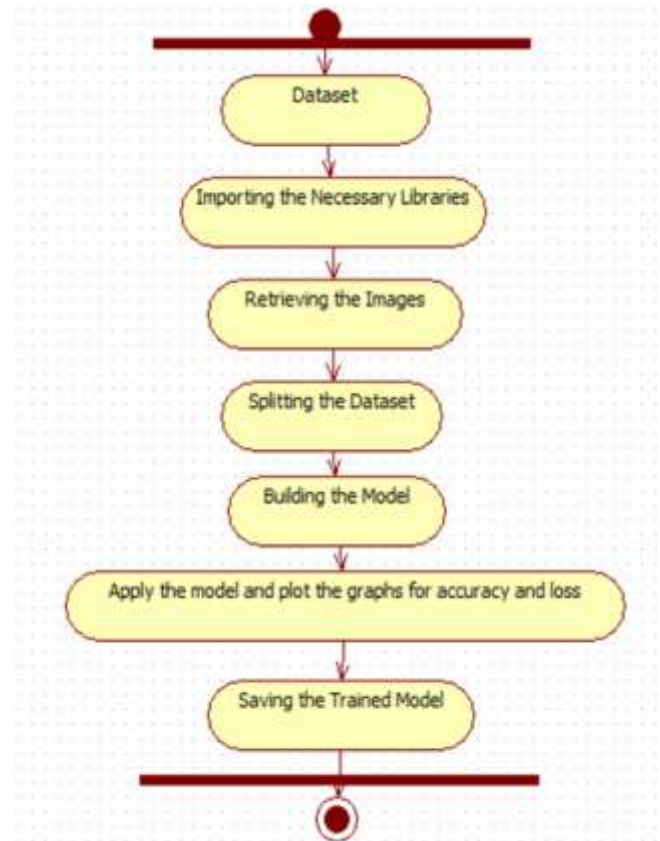
EXPLANATION: In the above diagram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

STATE DIAGRAM



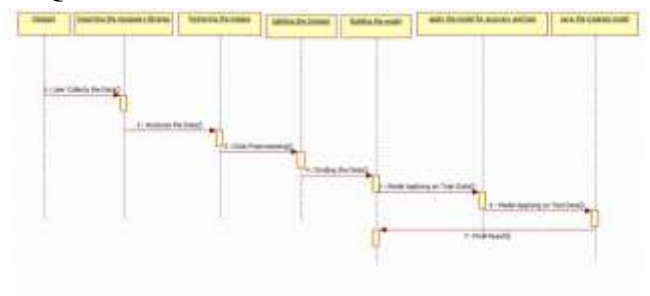
EXPLANATION: State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

ACTIVITY DIAGRAM



EXPLANATION: Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

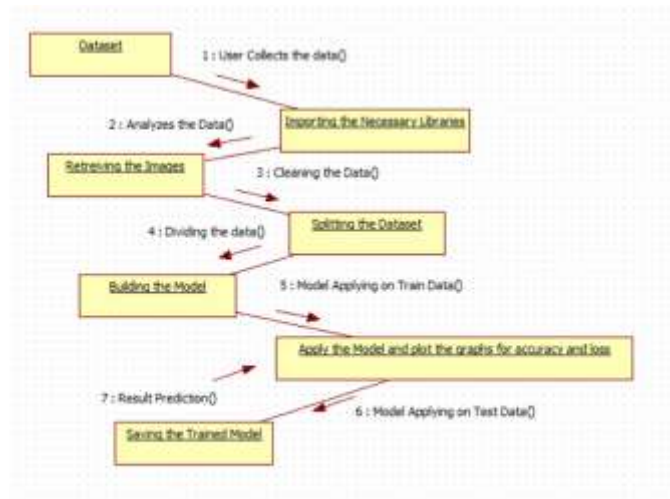
SEQUENCE DIAGRAM



EXPLANATION: A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object

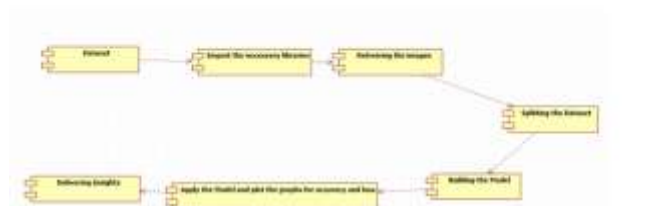
interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

COLLABORATION DIAGRAM



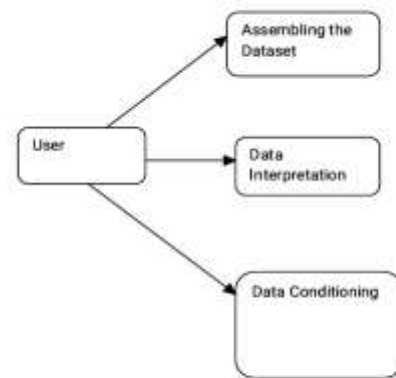
EXPLANATION: A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

COMPONENT DIAGRAM



EXPLANATION: In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

DATA FLOW DIAGRAM



Level 1

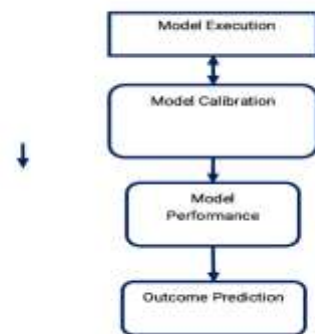
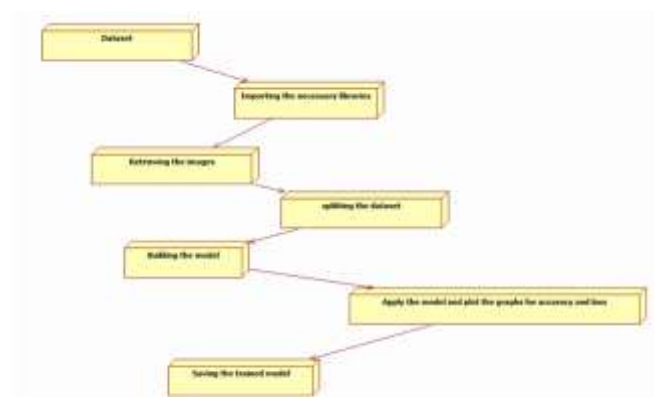


Fig 4.9: Data Flow Diagrams

EXPLANATION: Model Execution Model Calibration Model Performance Outcome Prediction Fig 4.9: Data Flow Diagrams A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

DEPLOYMENT DIAGRAM



EXPLANATION: Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

SYSTEM ARCHITECTURE

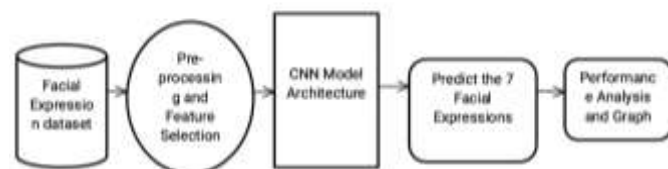


Fig 4.11: System Architecture

4.EXPERIMENTAL RESULTS

4.2 Results

Metric	value
Accuracy	98.52
Precision	95.2
Recall	94.8
F-Measure	95.9

Table -1: Model Performance Evaluation Results

The model achieves 998.52% accuracy on the test set, Following the detection of the sign by the system, a voice alert is sent through the speaker which notifies the driver

5. SYSTEM IMPLEMENTATION

The system is implemented using Python and deep learning techniques for real-time traffic sign recognition and voice alert generation. A Convolutional Neural Network (CNN) model is trained using the German Traffic Sign Recognition Benchmark (GTSRB) dataset containing multiple categories of traffic sign images. The images are preprocessed, resized, and converted into arrays before being fed into the CNN model for classification. The trained model is integrated with a Flask-based web application that captures input images, predicts the corresponding traffic sign, and generates voice alerts using the pyttsx3 library. The system provides accurate and fast recognition of traffic signs, helping drivers receive instant audio notifications for safer driving assistance.

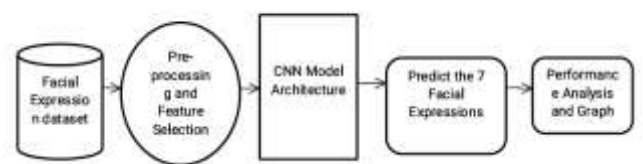


Fig -1: System Architecture

SOFTWARE TESTING

1.Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2.Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

3.System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

4.Performance Testing

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

5.Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

6.Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process

Build the test plan: Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

FUTURE ENHANCEMENTS

The prototype can be expanded to include an inbuilt alert system with a camera in the vehicle's centre. Also, the feature of getting the estimated time for reaching that particular traffic sign can be added. This system can also be expanded for identification of traffic signals and hence prompt the user about the time to reach that particular signal and its status as well. The user can accordingly plan their trip start time and hence cross all signals without having to wait. Also the driver verification will be done with the help of an API providing the information about the license holder and the license number.

HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented. • PROCESSOR • RAM • HARDDISK : : DUAL CORE2DUOS. 4GBDDR4 : 500 GB

SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity. • Operating System • Platform • Programming Language • Front End : : Windows 10 Spyder3 : Python Spyder

FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute based deduplication systems by resorting to the hybrid cloud architecture.

NON-FUNCTIONAL REQUIREMENTS

The major non-functional Requirements of the system are as follows

Usability The system is designed with completely automated process hence there is no or less user intervention.

Reliability The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

Performance This system is developing in the high level languages and using the advanced back-end technologies it will give response to the end user on client system with in very less time.

Supportability The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is built into the system.

Implementation The system is implemented in web environment using Jupyter notebook software. The server is used as the intelligence server and windows 10 professional is used as the platform. Interface the user interface is based on Jupyter notebook provides server system.

6. CONCLUSION

In this paper, our group proposes and designs a Traffic Sign Board Detection and Voice Alert System is implemented using Convolutional Neural Network. various models under the CNN heading were studied and the one with highest accuracy on the GTSRB dataset was implemented. The creation of different classes for each Traffic sign has helped in increasing the accuracy of the model. A voice message is sent after recognition of the sign which alerts the driver, thus helping him/her take appropriate decisions. This paper is a significant advancement in the field of driving as it would ease the job of the driver without compromising on the safety aspect. Also this system can easily be implemented without the need of much hardware thus increasing its reach.

ACKNOWLEDGEMENT

We express our sincere gratitude to our project guide for their valuable guidance, continuous support, and constructive suggestions throughout this research work on Traffic Sign Board Detection and Voice Alert System is implemented using Convolutional Neural Network, which greatly helped us improve the quality of our work. We also thank our faculty members and institution for providing the necessary resources, facilities, and environment to successfully complete this project. We acknowledge the support of open source tools and libraries such as Python, TensorFlow, and flask, as well as publicly available datasets that enabled efficient implementation and evaluation of the model. Finally, we extend our thanks to our team members and friends for their cooperation, effort, and support during all stages of this work, which played an important role in achieving the final results.

REFERENCES

- 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

BIOGRAPHY



Mr. Pathipati Chandrasekhar is currently serving as an Assistant Professor in the Department of Computer Science and Engineering at Guru Nanak Institute of Technology (Autonomous), Hyderabad. He earned his B.Tech from SYVSN Engineering College, affiliated with Nagarjuna University, and completed his M.Tech in Computer Science with First Class and Distinction from JINTU Kakinada, Andhra Pradesh. With over eight years of professional teaching experience in various engineering colleges across

Andhra Pradesh and Telangana, he has also worked for two years as a Cloud Consultant in Bangalore. He has published 11 research papers in national and international journals, attended 5 Faculty Development Programs (FDPs), and participated in 3 workshops. He is also the holder of 2 patents. His research interests include the Internet of Things (IoT), Artificial Intelligence, Quantum Computing, Deep Learning, and interdisciplinary applications.