AISSMS
COLLEGE OF ENGINEERING
Approved by AICTE, New Delhi, Recognized by
Govt. of Maharashtra, Affiliated to Savitribai Phule Pune University
and recognized 2(f) and 12(B) by UGC (Id.No. PU / PN/ Engg. / 093 (1992)
Accredited by NAAC with 'A+' Grade

A Project study Report on

Traffic Sign Recognition System

By

Mr. Aditya Mahale (21CO004)

Ms. Sharyu Adsul (21CO005)

Mr. Aniket Dhakane (21CO008)

Mr. Aryan Choudhary (21CO010)

Mr. Aryan Chouksey (21CO011)

Guide

Ms. M. M. Phadatare
Ms. Neha Rai

All India Shri Shivaji Memorial Society's College of Engineering, Pune, SE 2019 Pat.

ABSTRACT

This report presents the development of a Traffic Sign Recognition System (TSRS) using machine learning. The project involves training a convolutional neural network (CNN) model to accurately identify and classify traffic signs from input images. The model architecture includes convolutional, pooling, and fully connected layers, with dropout regularization for improved generalization. The model is trained using a dataset of labeled traffic sign images and optimized using the Adam optimizer. The system's performance is evaluated using metrics such as accuracy, and the trained model is deployed in a real-time system. The project contributes to enhancing road safety, driver assistance, and automation technologies.

TABLE OF CONTENT

Chapter No. 1

INTRODUCTION

1. TRAFFIC SIGN RECOGNITION

Traffic-sign recognition is a safety tech system that recognizes traffic signs and relays the information displayed on the sign to the driver through the instrument cluster, infotainment screen, or head-up display. Most TSR systems can identify speed limit, stop, and “do not enter” signs. More sophisticated systems may be able to recognize other types of signs.

The primary purpose of TSR is to increase driver focus. If a driver misses a sign, TSR can make them aware of it so they can react accordingly. The idea is simple: TSR identifies road signs the driver might have missed and alerts them of their presence.

This technology uses advanced forward-facing cameras positioned high on the windshield, generally adjacent to the rear-view mirror housing. Aimed to “see” traffic signs, the cameras scan the side of the road relative to the car.

Once the camera captures a sign, the system’s software processes the image to establish its classification and meaning. The system then relays this information to the driver almost instantaneously in the form of an icon or graphic representation of the sign. However, TSR’s ability to accurately identify a sign depends on the speed of the vehicle and its distance to the sign.

Chapter No. 2

LITERATURE REVIEW

- Traffic signs recognition with deep learning by Djebbara Yasmina, Rebai Karima, Azouaoui Ouahiba
- Real-Time Detection and Recognition of Road Traffic Signs by Jack Greenhalgh and Majid Mirmehdi
- Recognition of traffic signs based on their colour and shape features extracted using human vision models by X.W. Gao, K. Hong, L. Podladchikova, D. Shaposhnikov, N. Shevtsova
- Neural Networks and Deep Learning by Michael Nielsen

2.1 Problem Statement:

Traffic and street sign recognition by vehicles while driving and notification/warning the driver of same

2.2 Objectives:

1) To successfully classify traffic signs in categories to identify them.

2) To identify the traffic signs and correctly tell the name/meaning of a particular sign.

3) To demonstrate the project in a graphical interface to show the capability of the software.

2.3 Scope of project:

The scope of traffic sign recognition involves the detection, classification, and interpretation of traffic signs in real-time. It includes the ability to identify signs, classify them into specific categories, interpret their meaning, and operate effectively in various environmental conditions. Integration with other transportation systems and addressing limitations, such as challenging lighting or sign variations, are also part of the scope.

2.4 Methodology:

The following are the aspects to be taken into consideration for accomplishing our project:

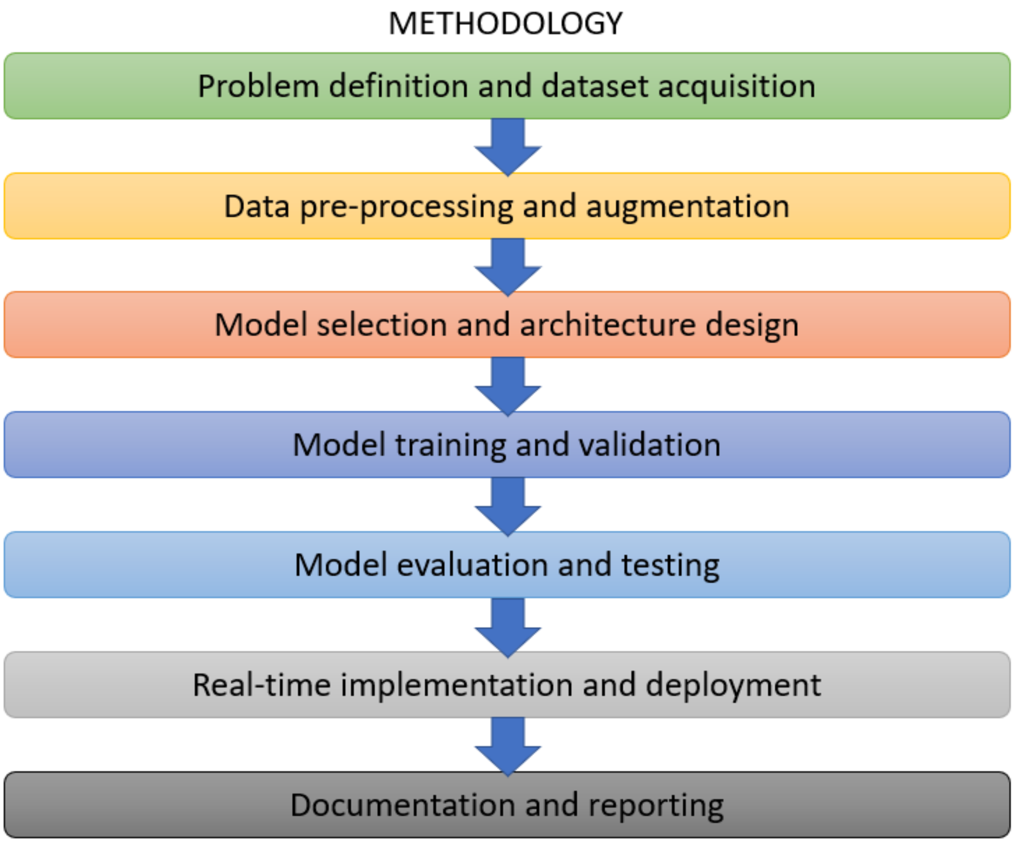
METHODOLOGY
Problem definition and dataset acquisition
Data pre-processing and augmentation
Model selection and architecture design
Model training and validation
Model evaluation and testing
Real-time implementation and deployment
Documentation and reporting

Fig. 2.1 – Methodology

2.5 Action Plan

Project Plan 2020-21											
Task to be completed		**Feb**	**March**				**April**				
		Week									
		1	1	2	3	4	1	2	3	4	
1	Searching of topic, Idea, case study etc	▨									
2	Literature review data collection work		🟦								
3	Topic finalization		🟦								
4	Defining objectives and expected outcomes			🟨							
4	Preparation of Action plan			🟨							
5	Methodology			🟨	🟨						
6	Design and calculations /Procedure (if any)				🟦	🟦					
7	Fabrication and testing of model(if any)						🟦	🟦			
8	Conclusion & Future scope								🟥		
9	Report preparation and submission								🟪	🟪	

Table 2.1 – Action Plan

Chapter No. 3

DESIGN AND CALCULATIONS

3.1 Design

1. Input Layer:
   - The input layer of the model is implicitly defined based on the shape of the training data (X_train).
   - The shape of X_train.shape[1:] indicates the input shape of the images (excluding the batch dimension).
2. Convolutional Layers:
   - The model starts with a Conv2D layer with 32 filters (output channels) and a kernel size of (5, 5).
   - The activation function used is ReLU (Rectified Linear Unit), which introduces non-linearity.
   - This layer performs convolution on the input images, applying the specified number of filters to extract features.
   - The resulting output shape is calculated based on the formula: (input_shape - kernel_size + 1) / stride.
   - In this case, the input shape is reduced by (5-1+1) = 5 in both dimensions, resulting in a new shape.
3. Max Pooling Layer:
   - The MaxPool2D layer with a pool size of (2, 2) reduces the spatial dimensions of the feature maps.
   - It selects the maximum value from each 2x2 region and outputs the maximum values only.
   - This reduces the computational complexity and introduces translational invariance.

- The output shape is calculated by halving the input shape in both dimensions.

4. Dropout Layer:

   - Dropout is a regularization technique that randomly sets a fraction of input units to 0 during training.
   - The Dropout layer with a rate of 0.25 is added to prevent overfitting by reducing interdependencies between neurons.

5. Additional Convolutional Layers:

   - Two more Conv2D layers with 64 filters and a kernel size of (3, 3) are added after the dropout layer.
   - They follow a similar structure to the initial convolutional layers but with different filter sizes.
   - The output shape is calculated using the same formula mentioned earlier.

6. Flattening Layer:

   - The Flatten layer reshapes the 2D output from the previous layer into a 1D vector.
   - This prepares the data for input to the fully connected (dense) layers.

7. Dense Layers:

   - A Dense layer with 256 units (neurons) and ReLU activation is added after the flattening layer.
   - This fully connected layer learns complex relationships between the extracted features.
   - Another Dropout layer with a rate of 0.5 is added to further prevent overfitting.

8. Output Layer:

   - The final Dense layer consists of 43 units (corresponding to the number of traffic sign classes).

- The activation function used is softmax, which calculates the probability distribution over the classes.
- This layer produces the final predictions for each traffic sign class.

3.2 Calculations:

The number of parameters in each Conv2D layer can be calculated using the formula: (filter_height * filter_width * input_channels + 1) * output_channels.

1) For the first Conv2D layer:

   ⇒ Number of parameters = (5 * 5 * input_channels + 1) * 32

2) For the second Conv2D layer:

   ⇒ Number of parameters = (5 * 5 * 32 + 1) * 32

3) For the third Conv2D layer:

   ⇒ Number of parameters = (3 * 3 * 32 + 1) * 64

4) For the fourth Conv2D layer:

   ⇒ Number of parameters = (3 * 3 * 64 + 1) * 64

5) For the Dense layer:

   ⇒ Number of parameters = (input_size * 256 + 1) * 256

6) For the final Dense layer:

   ⇒ Number of parameters = (256 * 43 + 1) * 43

The total number of parameters in the model can be obtained by summing up the parameters of all the layers.

Chapter No. 4

CONSTRUCTION & WORKING

Image Acquisition: The images are fed into the system for further processing.

Preprocessing: The uploaded images undergo preprocessing steps to enhance their quality and remove noise. This may involve techniques such as image filtering, color correction, and image resizing.

Traffic Sign Detection: Computer vision algorithms are employed to detect the presence and location of traffic signs within the captured images. These algorithms analyze visual features, such as color, shape, and texture, to identify potential sign candidates.

Feature Extraction: Once the signs are detected, features specific to each sign are extracted from the captured images. These features can include shape characteristics, color histograms, edge patterns, or other relevant attributes.

Classification and Recognition: The extracted features are then used to classify the detected signs into specific categories, such as speed limit signs, stop signs, or yield signs. Machine learning algorithms, such as convolutional neural networks (CNNs), are commonly utilized for this task. These algorithms are trained on large datasets of labeled sign images to learn the patterns and features indicative of different sign types.

Interpretation and Decision-making: After classifying the signs, the system interprets their meaning and provides relevant information. This can include conveying speed limits, regulatory instructions, or warning messages to drivers, autonomous vehicles, or traffic management systems.

Real-time Operation: The entire process, from image acquisition to sign detection, classification, and interpretation, is designed to operate in real-time. Efficient algorithms and hardware acceleration techniques are employed to ensure fast processing and timely responses.

Chapter No. 5
ADVANTAGES & DISADVANTAGES

5.1 Advantages

1) Improved road safety: By accurately detecting and interpreting traffic signs, the system can provide timely information to drivers, autonomous vehicles, and traffic management systems, leading to enhanced road safety and reduced accident risks.
2) Driver assistance: Traffic sign recognition can assist drivers by providing real-time alerts and warnings, helping them adhere to speed limits, navigate road regulations, and make informed decisions while driving.
3) Automation and efficiency: Integrating traffic sign recognition with autonomous vehicles enables them to understand and respond to traffic signs, contributing to safer and more efficient self-driving operations.
4) Enhanced traffic management: The data collected from traffic sign recognition systems can be used to monitor and analyze traffic patterns, optimize traffic flow, and improve overall transportation infrastructure planning and management.
5) Accessibility and inclusivity: Traffic sign recognition can support visually impaired individuals by converting visual traffic sign information into audible or tactile cues, making roads more accessible and inclusive.

5.2 Disadvantages

1) Environmental challenges: Adverse weather conditions, low lighting, occlusions, or complex urban environments can pose difficulties for traffic sign recognition systems, potentially affecting their accuracy and reliability.
2) Variability in sign designs: Traffic signs can exhibit variations in design, size, placement, and condition, which can impact the system's ability to detect and interpret signs consistently.
3) Computational requirements: Real-time processing and interpretation of traffic signs demand significant computational resources, which can pose challenges in terms of hardware requirements and processing speed.

4) Limited sign coverage: While traffic sign recognition systems aim to detect and interpret a wide range of signs, they may not cover all possible sign types or regional variations, potentially leading to missed or misinterpreted signs.

5) Human error and reliance: The system's effectiveness relies on accurate detection and interpretation, which may not be perfect. Human error in sign placement, damaged or vandalized signs, or the system misinterpreting certain situations can result in incorrect or unreliable information.

Chapter No. 6
APPLICATIONS

Applications:

1. Enhancing road safety: Traffic signs play a crucial role in guiding drivers and ensuring road safety. By developing a software system that can recognize and interpret traffic signs accurately, it can help drivers make informed decisions, reducing the risk of accidents and improving overall road safety.
2. Driver assistance and convenience: A traffic sign recognition system can serve as a driver assistance tool by providing real-time information about traffic signs. It can help drivers stay alert and comply with speed limits, no-entry signs, stop signs, and other important traffic regulations. This software can also enhance convenience by assisting drivers in unfamiliar areas, especially when signs are in foreign languages.
3. Traffic management and analysis: A traffic sign recognition system can provide valuable data for traffic management and analysis purposes. By automatically detecting and recording traffic signs, authorities can gain insights into traffic patterns, identify areas with inadequate signage, optimize road infrastructure, and implement targeted improvements to enhance traffic flow and safety.

Chapter No. 7

CONCLUSION

- In conclusion, the provided code implements a traffic sign recognition system using a convolutional neural network (CNN) approach. The code involves data preparation, model construction, training, evaluation, and saving. It assumes the availability of the German Traffic Sign Recognition Benchmark (GTSRB) dataset for training and evaluation.

- While the code is well-structured and covers essential components for traffic sign recognition, there are several considerations to keep in mind. The cost estimation for this project depends on factors such as development time, hardware and software requirements, training data availability, training time and resources, model evaluation needs, and ongoing maintenance.

- The cost estimation for the code implementation will vary based on your specific requirements, infrastructure, expertise, and any additional resources needed. It is essential to conduct a detailed analysis and consider factors such as development time, hardware expenses, data acquisition, training resources, and ongoing maintenance costs to obtain more accurate cost estimates.

- Overall, the code provides a foundation for building a traffic sign recognition system, but the cost estimation will depend on the specific project context and resource requirements. It is advisable to conduct a comprehensive assessment to determine the actual costs associated with implementing the system in your particular scenario.

Chapter No. 8
FUTURE SCOPE

The future scope of traffic sign recognition systems includes advancements in algorithms for improved accuracy and real-time performance. Integration with multi-modal sensor data, edge computing, and intelligent transportation systems can enhance functionality. Focus on generalization, privacy, and security is crucial. Dataset expansion, V2X communication integration, and augmented reality applications are also potential areas of development. Overall, the future holds opportunities for safer and more efficient traffic sign recognition systems.

REFERENCES

Websites:

https://www.sciencedirect.com/

https://en.wikipedia.org/wiki/Traffic-sign_recognition

https://data-flair.training/blogs/python-project-traffic-signs-recognition/

https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

Videos:

https://youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

https://www.youtube.com/watch?v=qahpZkPlTRM