A Project Report on

ADAPTIVE VISION

A Deep-Learning based Enhancement of Traffic Sign Detection For Autonomous Vehicles in Challenging Weather Conditions

in partial fulfillment for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

MARRI BALA SATISH (20B91A05H2)

Under the Guidance of

Sri S. Suresh Kumar

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM-534204, A.P
(Recognized by A.I.C.T.E New Delhi) (Accredited by NBA & NAAC)
(Affiliated to JNTU, KAKINADA)

[2023 – 2024]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

Chinna Amiram, Bhimavaram, West Godavari Dist., A.P.

[2023 – 2024]

BONAFIDE CERTIFICATE

This is to certify that the project work entitled **ADAPTIVE VISION – A Deep Learning based Enhancement of Traffic Sign Detection for Autonomous Vehicles in Challenging Weather Conditions** is the bonafide work of MARRI BALA SATISH (20B91A05H2) who carried out the project work under my supervision in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

SUPERVISOR
(S. Suresh Kumar)
Assistant Professor

HEAD OF THE DEPARTMENT
(Dr. V Chandra Sekhar)
Professor

SELF DECLARATION

I hereby declare that the project work entitled **ADAPTIVE VISION – A Deep Learning based Enhancement of Traffic Sign Detection for Autonomous Vehicles in Challenging Weather Conditions** is a genuine work carried out by us in B.Tech., (Computer Science and Engineering) at SRKR Engineering College(A), Bhimavaram and has not been submitted either in part or full for the award of any other degree or diploma in any other institute or University.

Marri Bala Satish (20B91A05H2)

TABLE OF CONTENTS

ABSTRACT

Autonomous vehicles heavily rely on accurate and timely detection of traffic signs for safe navigation. In the process of traffic sign detection, the size of the objects and weather conditions exhibit significant variability, which can influence the detection accuracy. This project aims to enhance the robustness in detecting the traffic signs in challenging weather conditions through the implementation of the deep learning techniques.

Adverse weather conditions pose significant challenges for autonomous vehicles (AVs), particularly in rain, snow, and similar scenarios. To address these challenges, we introduce a deep learning system designed for the enhancement of the detection of traffic signs, providing a clear understanding for Autonomous vehicles. Our proposed system aims to optimize traffic sign board detection, enabling reliable performance in all weather conditions. Through the utilization of deep learning techniques, we seek to unlock the potential for all-weather AV navigation, contributing to safer and more robust autonomous driving experiences.

LIST OF TABLES

LIST OF FIGURES

CHAPTER - 1
INTRODUCTION

In the domain of autonomous vehicles, the ability to accurately perceive and interpret traffic signs under varying environmental conditions is essential for ensuring safe navigation. However, traditional computer vision algorithms often struggle in challenging weather conditions or low-light situations, hindering reliable traffic sign detection and compromising the safety of autonomous driving systems. To address this challenge, our project, titled "Enhancing Traffic Sign Detection for Autonomous Vehicles in Challenging Weather Conditions through Deep Learning," aims to develop an advanced solution. Our approach involves utilizing deep learning techniques to significantly improve traffic sign detection performance.

The rise of autonomous vehicles marks a significant transformation in transportation systems worldwide. An essential aspect of their safe operation is their ability to detect and interpret traffic signs accurately in real-time. However, challenging weather conditions pose a significant obstacle to this objective. Traditional computer vision algorithms often struggle to maintain reliable performance when faced with reduced visibility, changes in lighting, and occlusions caused by variability in weather conditions. Consequently, ensuring consistent traffic sign detection under such challenging circumstances remains a crucial hurdle for the widespread adoption of autonomous driving systems.

Our project aims to develop an adaptive vision system capable of reliably detecting traffic signs across diverse weather conditions by leveraging deep learning techniques. Through convolutional neural networks (CNNs) and recurrent neural networks (RNNs), we aim to extract detailed features from traffic sign images, enabling the system to generalize effectively across different weather conditions and lighting scenarios.

The significance of our project extends beyond autonomous vehicles. By enhancing the perceptual capabilities of automated driving systems, our solution has the potential to improve safety, efficiency, and reliability in transportation networks globally. Furthermore, the methodologies and techniques developed through this project offer valuable insights into addressing challenges related to adverse environmental conditions in various applications beyond autonomous driving.

In conclusion, our project represents an innovative effort to tackle the critical challenge of traffic sign detection in adverse weather conditions using different deep learning techniques. Through this endeavor, we aim to contribute significantly to the advancement of autonomous driving technology, ultimately paving the way for safer and more efficient transportation systems in the future.

CHAPTER - 2

LITERATURE SURVEY

Author	Approach used	Dataset	Accuracy	Drawback
Qu S, Yang X [2]	YOLOv5	CCTSDB 2021	88.1%	Less accuracy
Cire san et al. [8]	MCDNN	GTSRB[a]	99.46%	Poor results for blurred images
Yuan et al. [9]	VSSA-net	VOC 2007	94.6%	Approach limited to small datasets
Shu-Chun Huang [10]	Alexnet	GTSDB[b]	92.63%	Cannot distinguish the background from the traffic sign
Zhongqin Bi [11]	Improved VGG	GTSRB	98.47%	Poor environmental conditions not considered
Cen Han [12]	Faster RCNN	Self	92.00%	Greater detection time
Chuanwei Zhang [13]	Lenet-5	GTSDB	94%	Slower response time
Zhao Wang [14]	CNN	GTSRB	95.0%	Poor environmental conditions not considered
Chengji Liu [15]	YOLO[c]	GTSDB	94%	Slower response time
Smit Mehta [16]	CNN	BTSD[d]	98.26%	Poor results for blurred images
Choy Ja Yeong [17]	Raspberry Pi	GTSRB	90.00%	Greater detection time

Table 1: Literature survey summary

- [2] In the work of Qu and Yang, to develop a YOLOv5 model, the researchers applied the CCTSDB 2021 dataset in the training process. However, their model achieved a relatively low accuracy level of 88.1%, which complicates its application in practice. Additionally, the selected dataset does not represent the standard benchmark for the target task.

- In the work, “Multi - column deep neural network for traffic sign classification”, Cire et al. developed and explored the performance of the MCDNN model using the GTSRB dataset. With the impressive accuracy of 99.46%, the model’s performance was significantly high. Nevertheless, it underperformed when provided with the blurred images, and it is necessary to investigate the implications of blurry visual data for the performance of the original and similar models. In the process, the existing solutions cannot be considered as the optimal choice due to the above-mentioned factor and it is required to conduct new research and create a new efficient and smart system.

- In Ref. [9], the authors introduced the VSSA-net model and evaluated its performance using the VOC 2007 dataset. According to the results, the proposed approach could achieve an accuracy of 94.6%. However, it was also noted that the model developed by Yuan et al. demonstrated low performance on large datasets. Thus, further research should be conducted to improve the scalability and generalization of the proposed approach for small datasets.

- Shu-Chun Huang has experimented the AlexNet architecture on the GTSDB dataset and obtained a good result of 92.63% in terms of accuracy. The drawback of this approach is that it failed to well distinguish actual traffic signs from the background; this suggests that the feature extraction and discrimination mechanisms must be upgraded to improve the accuracy and reliability of the used solution for real-world traffic sign recognition.

- [11] An upgraded version of the VGG model was presented by Zhongqin Bi and its performance on the GTSRB dataset evaluated with an impressive accuracy of 98.47%. But in the study, they never mentioned anything about how it would work under harsh weather conditions. In other words, this can be a limitation that shows why further

research is necessary so as to improve the toughness of the model and make it suitable for different environmental settings including those that are unfavorable.

- Cen Han applied the Faster R-CNN model and assessed its effectiveness on his personal data set, resulting in 92.00% accuracy that was quite good. However, one weakness of this method is the increased detection time as against other models which indicate a balance between efficiency and accuracy. This highlights the importance of further optimization to enhance the speed of the model while keeping its detection performance for real-time applications helpful [12].

- Chuanwei Zhang employed the LeNet-5 architecture and assessed its performance on the GTSDB dataset, achieving a respectable accuracy of 94% [13]. However, a notable limitation of this implementation was its slower response time, indicating a potential trade-off between accuracy and speed. This highlights the importance of optimizing the model's architecture or utilizing hardware acceleration techniques to improve its real-time performance for time-sensitive applications.

- Zhao Wang's work involves the usage of a CNN model, which was evaluated on one database, the GTSRB database, with an accuracy of 95.0%. The technique did not take into account the tough environmental circumstances found in real-world situations reported in [14]. As a result of this neglect, the model's flexibility and generalizability may suffer. As a result, future research should focus on addressing this issue in order to improve the practical use of models under a variety of environmental conditions.

- [15] Chengji Liu utilized the YOLOc (You Only Look Once) model and evaluated its performance on the GTSDB dataset, achieving a commendable accuracy of 94%. However, a drawback of this implementation was its slower response time, indicating a compromise between accuracy and speed. This highlights the need for further optimization to enhance the model's efficiency without sacrificing its detection accuracy, particularly for real-time applications where swift processing is crucial.

- Smit Mehta implemented a CNN model and evaluated its performance on the BTSD dataset, achieving a high accuracy of 98.26%. However, the model exhibited poor performance when presented with blurred images, indicating a limitation in its ability to effectively handle such scenarios. This underscores the importance of enhancing the

model's robustness to handle diverse and challenging image conditions, including blurred images, for improved real-world applicability [16].

- [17] Choy Ja Yeong utilized a Raspberry Pi-based system for traffic sign recognition and evaluated its performance on the GTSRB dataset, achieving an accuracy of 90.00%. However, one notable drawback of this implementation was the greater detection time compared to other systems or models. This suggests a trade-off between accuracy and processing speed on the Raspberry Pi platform, highlighting the need for optimization strategies to improve real-time performance while maintaining satisfactory accuracy for practical applications.

CHAPTER – 3

OBSERVATIONS FROM LITERATURE SURVEY

- Within our in-depth exploration of the present research for the project titled "Enhancing Traffic Sign Detection in Harsh Weather Conditions for Autonomous Vehicles Using Deep Learning", we have managed to identify some crucial insights that stemmed from past studies.

- Notably, research papers have employed various deep learning models such as R-CNN, customized CNN, older versions of YOLO (v2, v3, v5) architectures to tackle the challenges of traffic sign detection. While some studies have reported promising results, there's a consensus on the difficulties posed by adverse weather conditions. A subset of these investigations has shed light on the limitations associated with small datasets, leading to decreased accuracy levels.

- These findings emphasize the critical need to expand datasets and explore more sophisticated deep learning architectures. This literature survey lays the groundwork for our project, highlighting the importance of addressing existing challenges through innovative approaches and advancements in the field.

- We've noted a common concern: the demanding nature of object detection in autonomous driving scenarios. While our literature review has uncovered valuable insights, we acknowledge a prevailing gap in the consideration of occlusion, a critical factor in real-world applications.

Existing systems also mentioned that the framework can be improved through data augmentation and object persistence.

Key points to be considered for further implementation:

- Using large and benchmark dataset(s) for good accuracy.
- Considering the situation of Object Occlusion.
- Improving the framework with data augmentation and object persistence.

CHAPTER - 4

PROBLEM STATEMENT

In adverse weather conditions, conventional computer vision often struggle to accurately detect traffic signs due factors like poor visibility, altered lighting, and obstructions such as raindrops, snowflakes, or fog. This deficiency poses a significant safety risk for autonomous vehicles, as reliable traffic sign detection is crucial for making critical driving decisions promptly and effectively. Existing solutions primarily rely on handcrafted features and rule-based algorithms, which lack the adaptability and resilience needed to cope with the diverse challenges posed by adverse weather conditions.

To address this critical limitation, our project seeks to pioneer a transformative approach by leveraging advanced deep learning algorithms. By harnessing the power of deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), we aim to develop a robust and adaptable traffic sign detection system capable of operating effectively across a spectrum of challenging weather conditions. Through this endeavour, we aspire to significantly enhance the safety and reliability of autonomous vehicles, ushering in a new era of intelligent transportation systems

CHAPTER - 5

METHODOLOGY

5.1 PROPOSED SOLUTION

In response to the identified drawbacks in previous research, our proposed project aims to address these challenges comprehensively.

- We intend to mitigate limitations associated with small datasets by incorporating benchmark datasets, fostering increased diversity and scale. Employing innovative techniques like data augmentation will further augment our dataset, enhancing the model's ability to generalize and improving overall accuracy.

- To specifically tackle the adverse weather conditions that pose challenges to traffic sign detection, our approach involves considering a diverse range of weather scenarios during both training and testing phases.

- In addition, we plan to leverage advanced deep learning models, including Convolution Neural Network (CNN), YOLOv8, ResNet50, and others, to elevate the sophistication and efficiency of our traffic sign detection system.

Through these strategic measures, our project aims to significantly enhance the robustness and performance of traffic sign detection in adverse weather conditions for autonomous vehicles, contributing to the advancement of safety and reliability in autonomous navigation.

5.2 PROPOSED IMPLEMENTATION

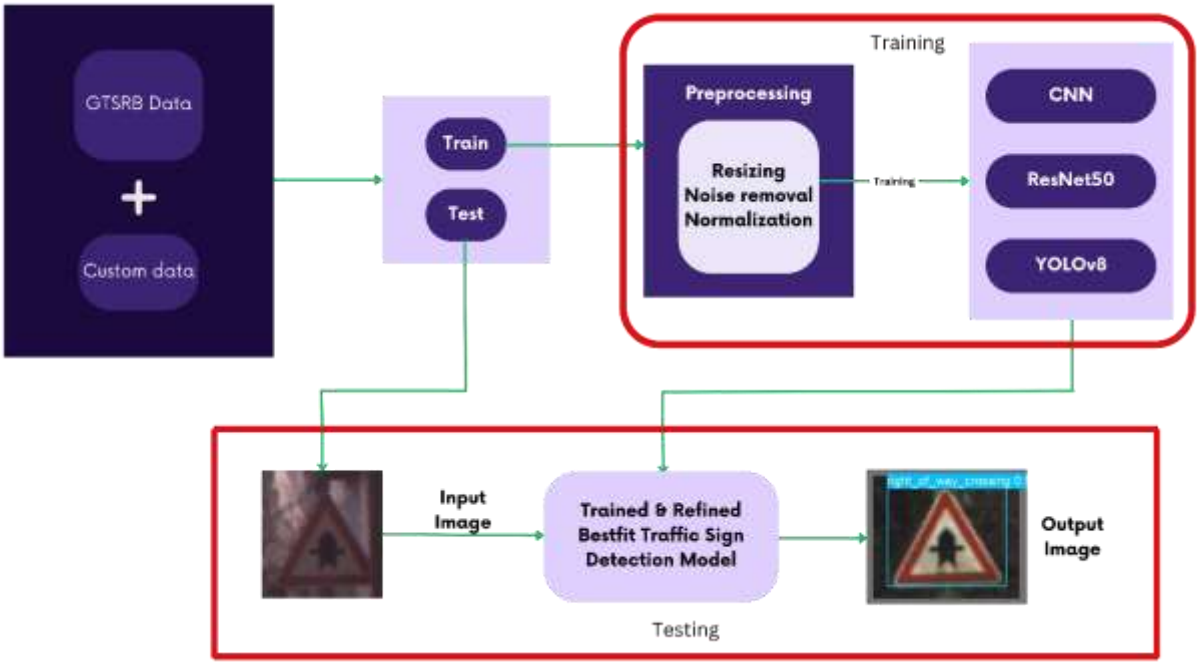


Fig. 1: Proposed Implementation

- **Video Capture:** HD cameras mounted on autonomous vehicles capture video footage, sampled at specific frame rate. Environmental conditions enhance image quality to provide clear input for subsequent processing stages.

- **Preprocessing:** Captured video undergoes preprocessing to enhance image quality and prepare for analysis. Different techniques like Median filtering, Lucy Richardson filter and Histogram equalization etc., are used for removing noise from images.

- **Traffic Sign Recognition & Detection:** Recognition & Detection of traffic signs must be quick and efficient to ensure proper functioning of the system so, algorithms like CNN, YOLOv8, ResNet50 etc., are employed for real-time detection of traffic signs.

CHAPTER - 6

DATASET OVERVIEW

- The German Traffic Sign Recognition Benchmark (GTSRB) dataset is a widely used dataset in the field of computer vision, especially in the field of traffic sign recognition. It is designed to develop and evaluate algorithms for the task of classifying traffic signs commonly found on roads.
- The German Traffic Sign Recognition Test (GTSRB) includes 43 types of traffic signs, divided into 39,209 training images and 12,630 test images.
- Images feature different lighting conditions and rich backgrounds.

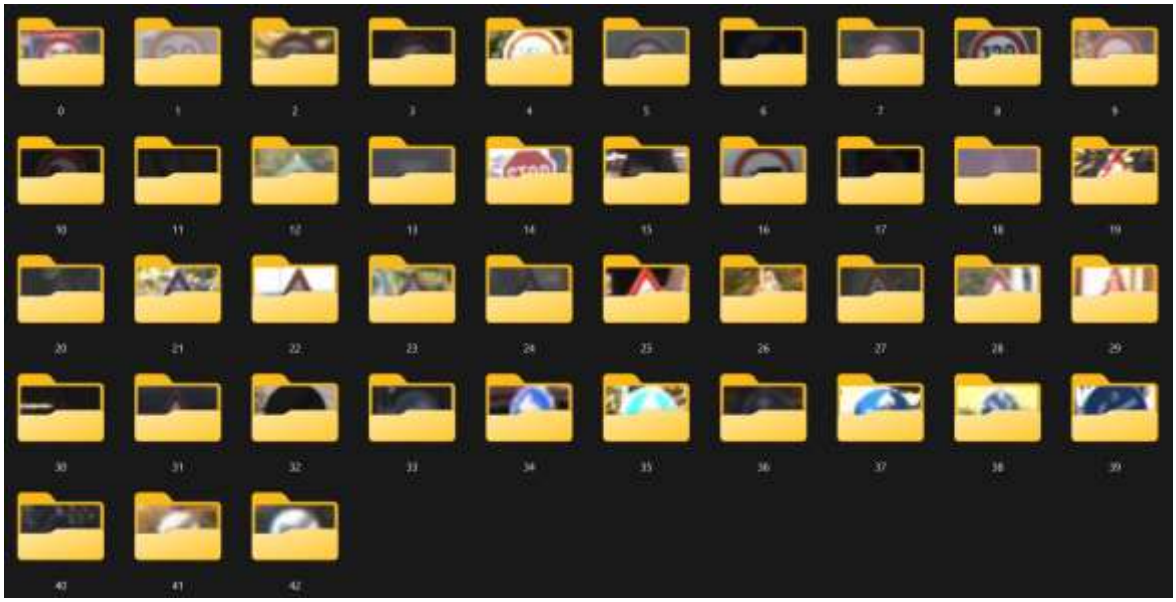


Fig. 2: Dataset Overview

- Different classes of GTSRB dataset:

1. Speed limit (20 km/h)
2. Speed limit (30 km/h)
3. Speed limit (50 km/h)
4. Speed limit (60 km/h)
5. Speed limit (70 km/h)
6. Speed limit (80 km/h)
7. End of speed limit (80 km/h)
8. Speed limit (100 km/h)
9. Speed limit (120 km/h)
10. No passing

11. No passing for vehicles over 3.5 metric tons
12. Right-of-way at the next intersection
13. Priority road
14. Yield
15. Stop
16. No vehicles
17. Vehicles over 3.5 metric tons prohibited
18. No entry
19. General caution
20. Dangerous curve to the left
21. Dangerous curve to the 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 of all speed and 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 of no passing by vehicles over 3.5 metric tons

- Number of samples per class:

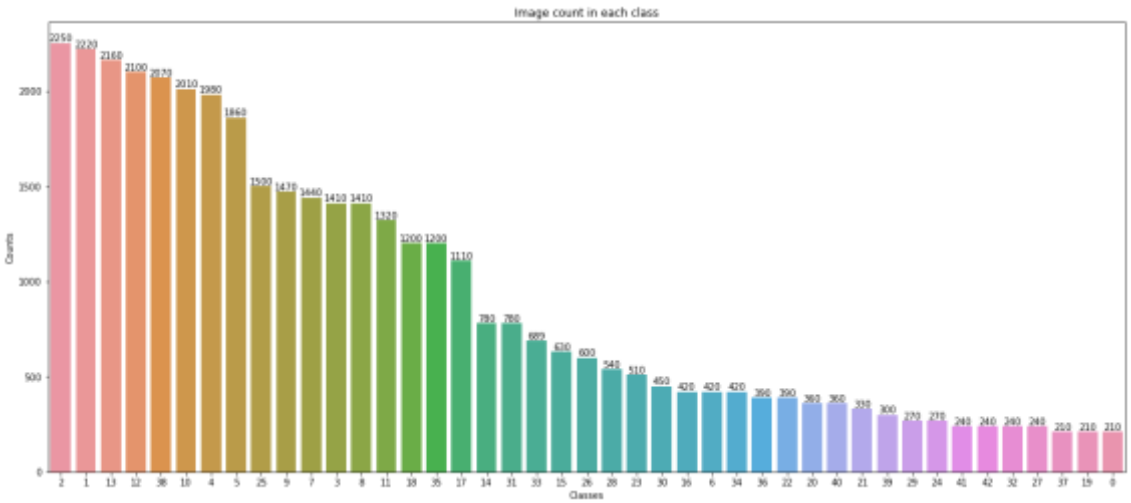


Fig. 3: Distribution of different classes in dataset

- Sample images from GTSRB dataset:

Fig. 4: Sample images of Dataset

Reference: Dataset Link

CHAPTER - 7

IMPLEMENTATION

- As a part of our research project, we have implemented various Machine Learning and Deep Learning models to get the most accurate one.
- We have trained the models with a training data of 39,209 images and had done the testing and validation on 12,630 test images which in whole contain 43 different classes of traffic signs.
- The implemented models are:
    1. Convolution Neural Network (CNN)
    2. Residual Network (ResNet50)

7.1 CONVOLUTION NEURAL NETWORK (CNN)

CNN Architecture

- The term "CNN" stands for Convolutional Neural Network. It's a type of artificial neural network, particularly suited for processing structured grid-like data, such as images.
- Different layers of CNN architecture used in this model are:
    - Input Layer: Accepts images with dimensions (IMG_HEIGHT, IMG_WIDTH, channels), where IMG_HEIGHT and IMG_WIDTH represent the image dimensions, and 'channels' represents the number of color channels (e.g., 3 for RGB).
    - Convolutional Layers: Consist of two pairs of Conv2D layers. Each pair has two Conv2D layers with increasing numbers of filters (16 in the first pair, 32 and 64 in the second pair) and ReLU activation functions.
    - MaxPooling Layers: Follow each pair of convolutional layers to downsample the feature maps using max-pooling with a pool size of (2, 2).

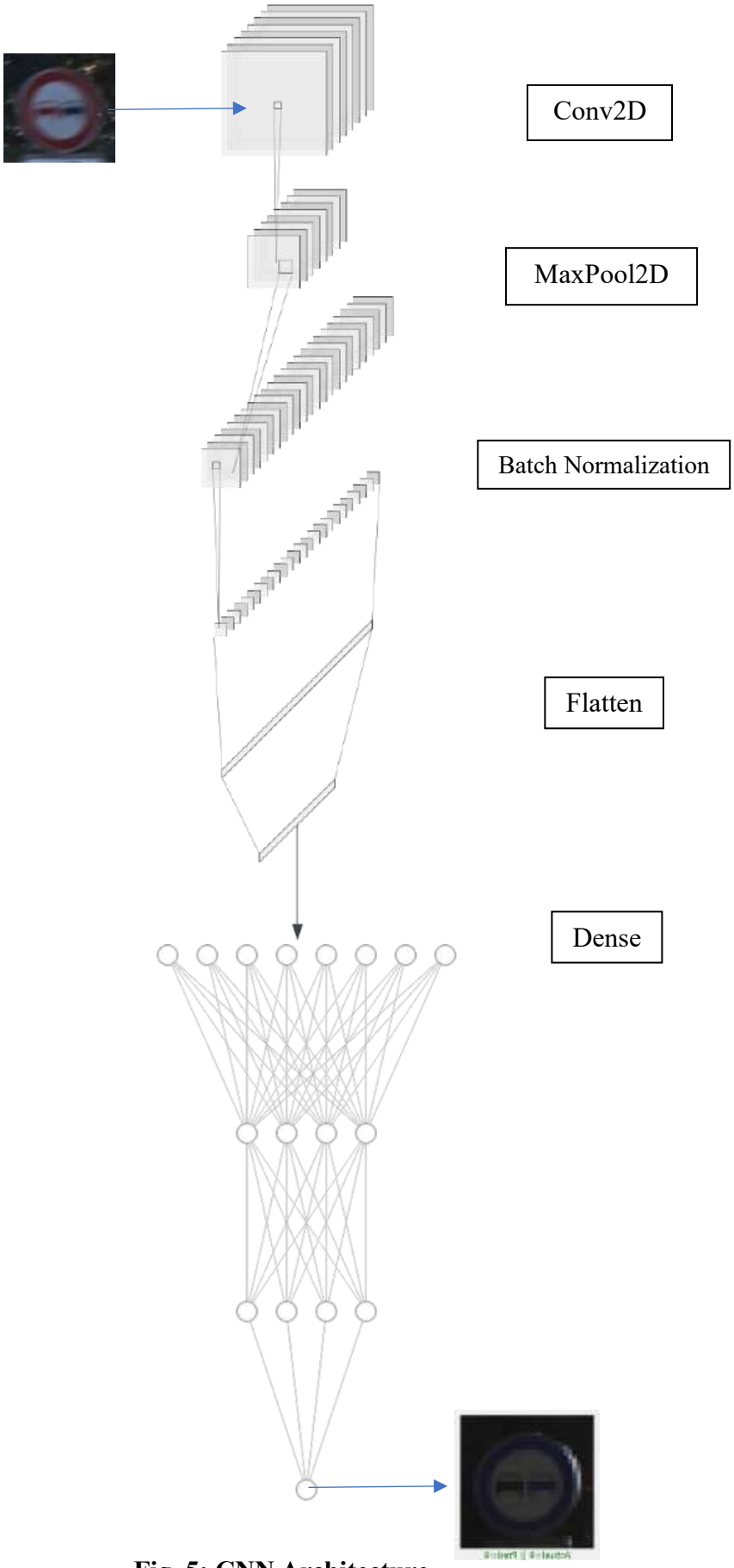


Fig. 5: CNN Architecture

- Batch Normalization: These layers are added after each pair of convolutional layers to normalize activation, improving stability and training speed.
- Flatten Layer: Converts 2D feature maps to 1D feature vectors, which will be fed into fully connected layers.
- Fully Connected Layers: Includes a dense layer with 512 units and ReLU activation function. Batch normalization is typically applied before the dropout layer with a dropout rate of 0.5, to reduce overfitting and improve model generalization.
- Output Layer: Consists of a dense layer with 43 units (equal to the number of layers) and a softmax activation function, which generates a prediction probability for each layer.

Overall, this CNN architecture is designed for multi-class classification tasks, where the goal is to classify the input image into one of 43 classes. The model uses convolutional layers for feature extraction, max pooling layers for spatial reduction, and fully connected layers for classification. Normalization and batch cancellation are integrated to improve training stability and reduce overfitting.

7.2 RESIDUAL NETWORK (ResNet50)

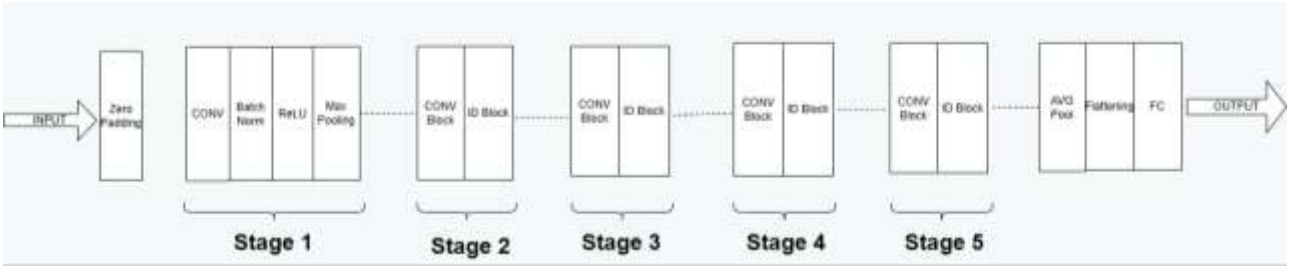


Fig. 6: ResNet50 Architecture

ResNet50 Architecture

- ResNet50 is a convolutional neural network (CNN) architecture that was introduced by Kaiming He et al. in their paper titled "Deep Residual Learning for Image Recognition" in 2015. It is part of the ResNet50 (Residual Network 50) family of models, which are renowned for their effectiveness in training very deep neural networks.
- Here are the different layers of the ResNet50 model used in the code:
    - Input Layer: The input shape in the ResNet50 model is defined as (32, 32, 3), indicating that it expects images with a size of 32x32 pixels to have three color channels (RGB).
    - Convolutional Layers ResNet50 includes a range of convolutional layers, involving convolutional blocks containing residual connections. They extract keys from the input images at various abstraction levels.
    - Max Pooling Layers: Utilizing max pooling layers, ResNet50 aims to downsample the feature maps produced by the convolutional layers. Max pooling helps minimize the spatial dimensions of the feature maps while still preserving significant information.
    - Fully Connected Layers: The fully connected layers, also recognized as the dense layers, are not utilized straightforwardly in the presented code. Rather than that, the output from the last convolutional layer of ResNet50 is flattened (Flatten()) to establish a one-dimensional array, which is subsequently handed over to the custom dense layers implanted on top for classification.

- Base Model (ResNet50): The ResNet50 model itself containers many layers, including convolutional layers, pooling layers, and residual blocks. However, as it's instantiated with include_top = False, only the convolutional layers are added to the model. The number of layers in the ResNet50 base model can be obtained using the summary() method.
- Flatten Layer: After the ResNet50 base model, a Flatten layer is added, which does not add any new trainable parameters. It simply flattens the output of the ResNet50 base model into a one-dimensional array.
- Dense Layers: Two dense layers are added on top of the Flatten layer. The first dense layer containers 512 units, and the second dense layer contains 43 units (the number of classes in the dataset).
- Dropout Layer: A Dropout layer is a rare bird, with a dropout rate of 0.5, added after the first dense layer.

- In short, this ResNet50 architecture does not directly manipulate or modify the internal layers of the ResNet50 model. Instead, it leverages the pre-trained ResNet50 model as a fixed feature extractor, extracting relevant features from the input image, which are then fed into custom dense layers for classification purposes.

7.3 You Only Look Once (YOLO)

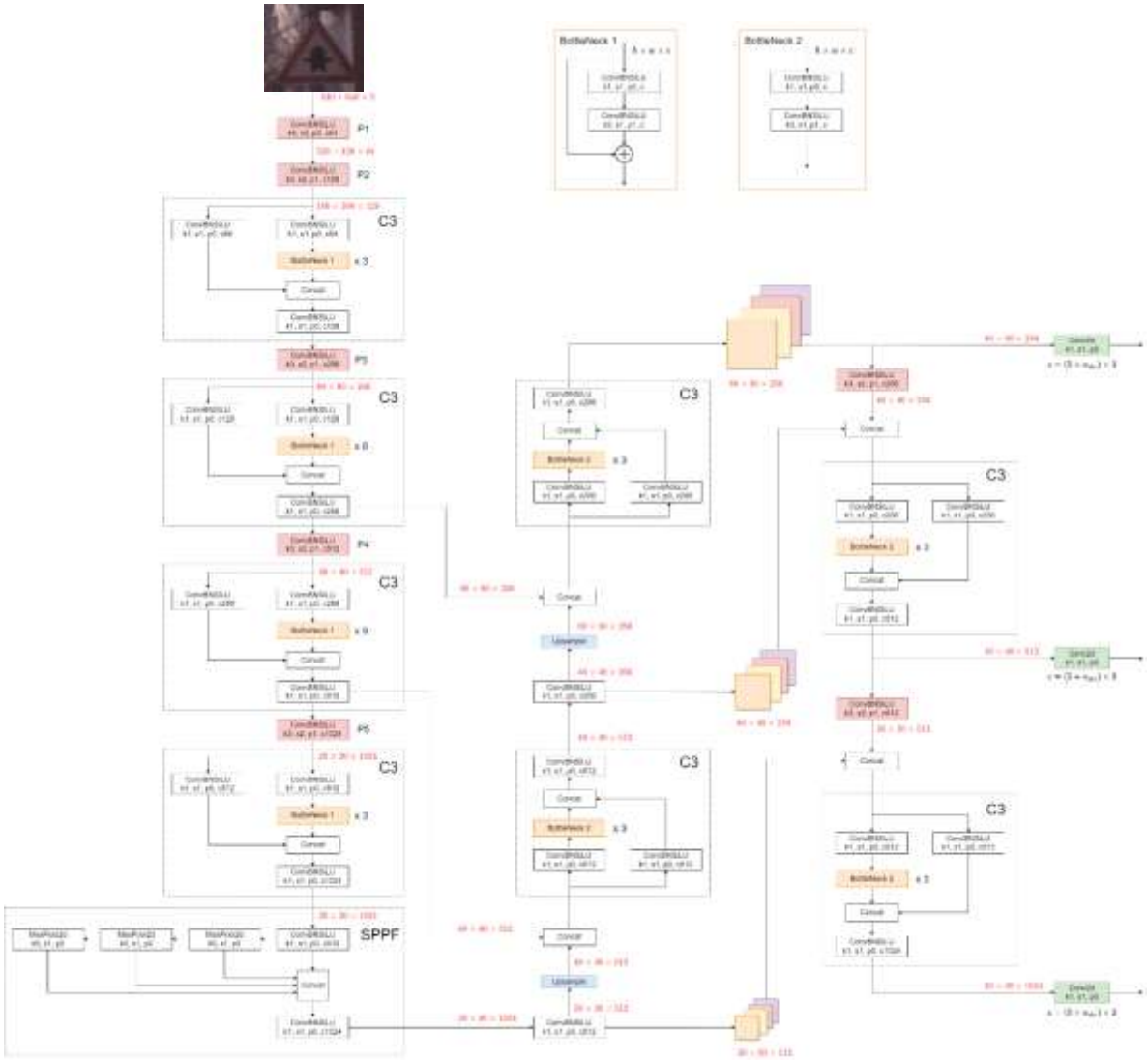


Fig. 7: YOLOv8 Architecture

YOLOv8 Architecture

- YOLO, short for "You Only Look Once," is a state-of-the-art object detection algorithm that revolutionized computer vision tasks. Unlike traditional methods that require multiple passes over an image, YOLO applies a single neural network to the entire image, dividing it into grid cells and predicting bounding boxes and class probabilities for each cell simultaneously. This results in real-time detection with impressive speed and accuracy, making it a popular choice for various applications, including autonomous driving, surveillance, and object tracking.
- YOLOv8, an evolution of the You Only Look Once (YOLO) object detection algorithm, represents a significant advancement in real-time object detection. Building upon

previous versions, YOLOv8 incorporates improvements in architecture, training techniques, and model optimization to achieve superior performance.

- Different layers of YOLOv8 architecture are:
    - Input Layer: This layer receives the input image and passes it through the network for processing.
    - Backbone (Darknet53 or CSPDarknet53): The backbone is responsible for feature extraction from the input image. In YOLOv8, it typically employs a deep convolutional neural network architecture like Darknet53 or CSPDarknet53 to extract hierarchical features at different scales.
    - Neck: The neck, often referred to as the feature pyramid network (FPN), enhances feature representation by aggregating features from different layers of the backbone network. This helps in capturing both high-level and low-level features necessary for accurate object detection.
    - Detection Head: This component consists of several convolutional layers responsible for predicting bounding boxes, objectness scores, and class probabilities for detected objects. YOLOv8 typically predicts bounding boxes at multiple scales to handle objects of different sizes more effectively.
    - Output Layer: The output layer provides the final predictions generated by the detection head, including bounding box coordinates, confidence scores, and class probabilities for each detected object.
    - Loss Function: YOLOv8 utilizes a specific loss function, often a combination of localization loss (such as smooth L1 loss) and classification loss (commonly implemented using softmax or sigmoid activation functions), to optimize the network during training. The loss function measures the disparity between predicted and ground truth bounding boxes and class labels.
    - Non-Maximum Suppression (NMS): After predictions are generated, YOLOv8 typically applies NMS to remove redundant or overlapping bounding boxes, retaining only the most confident detections.

CHAPTER - 8

PERFORMANCE EVALUATION & RESULT ANALYSIS

S.No.	Model	Evaluation Metrics
1.	CNN	Accuracy: 0.98 Weighted Precision: 0.97 Weighted F1-score: 0.98 Weighted Recall: 0.95 No. of Epochs: 30
2.	ResNet50	Test Accuracy: 0.93 Test Loss: 0.43 Precision: 0.91 Recall: 0.90 F1-score: 0.90 No. of Epochs: 100
3	YOLOv8	Accuracy: 0.94 Precision: 0.91 Recall: 0.89 F1-score: 0.89

Table 2: Performance metrics

8.1 CNN Model METRICS

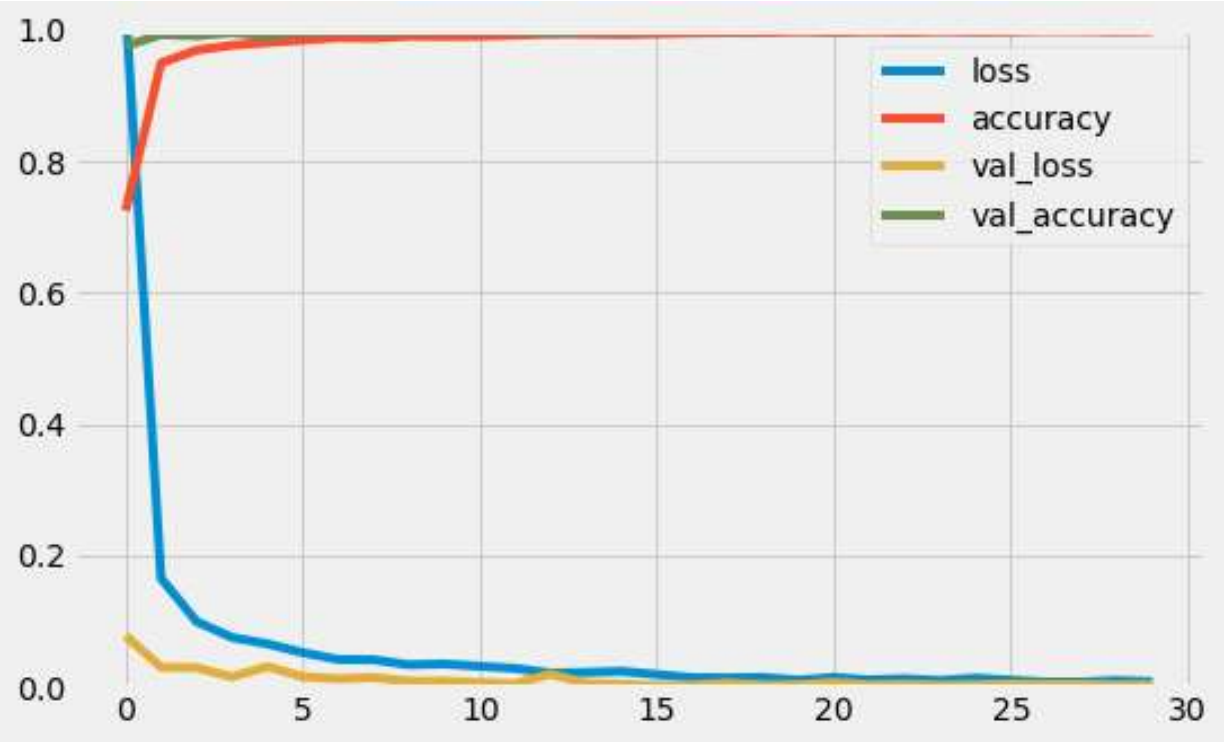


Fig. 8: CNN Model Metrics

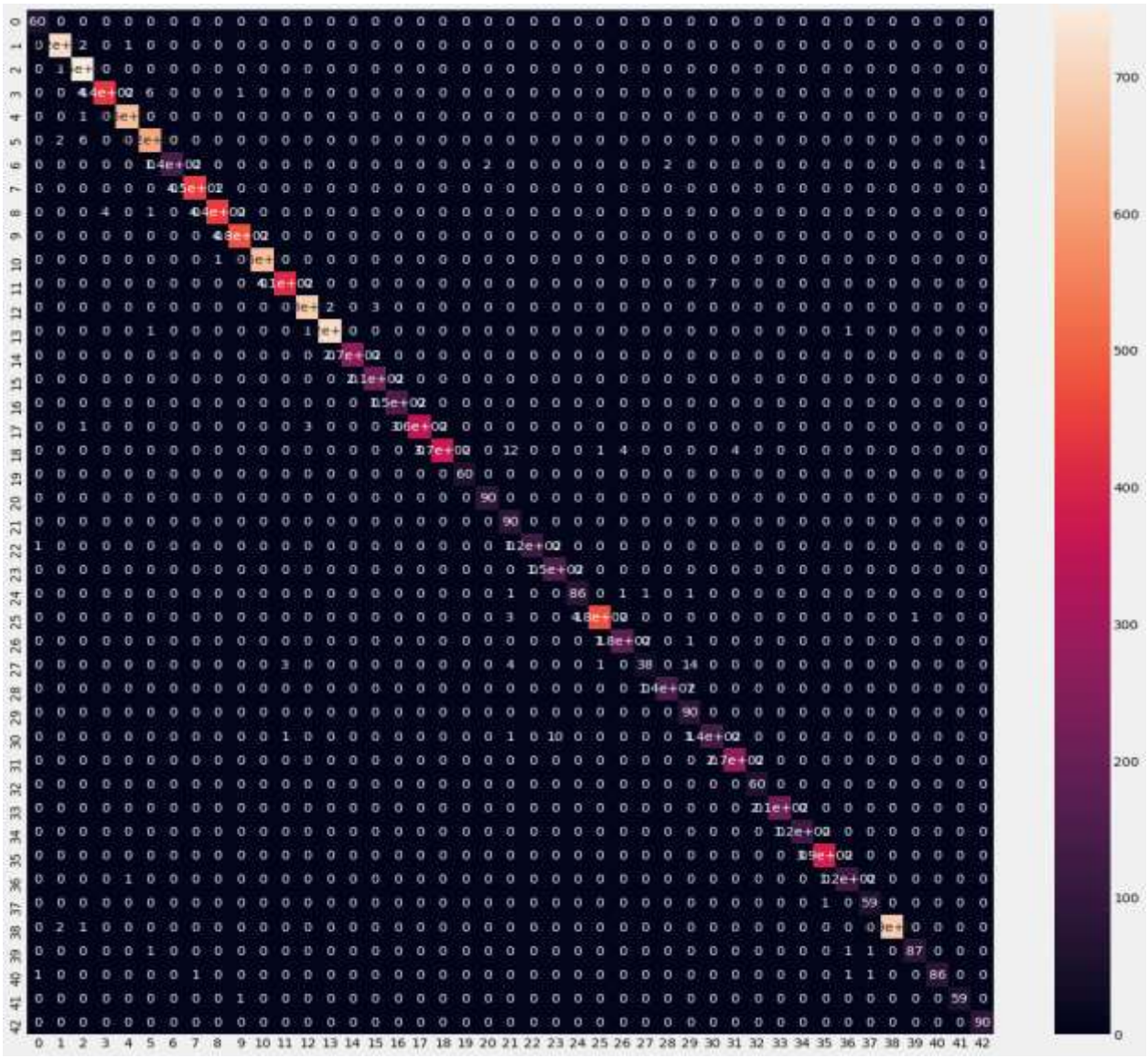


Fig. 9: CNN model Heatmap

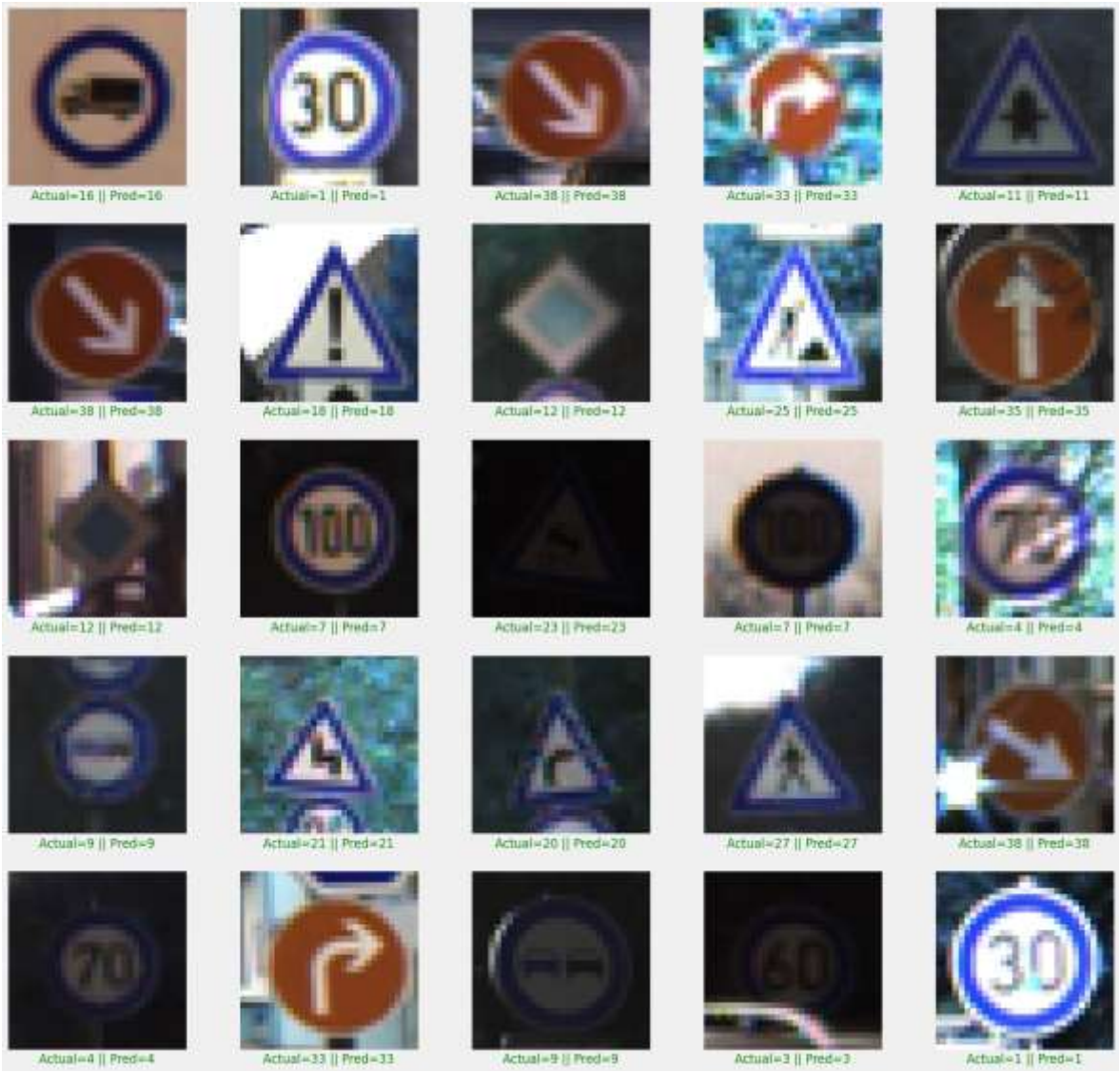


Fig. 10: CNN model output

8.2 ResNet50 Model METRICS:

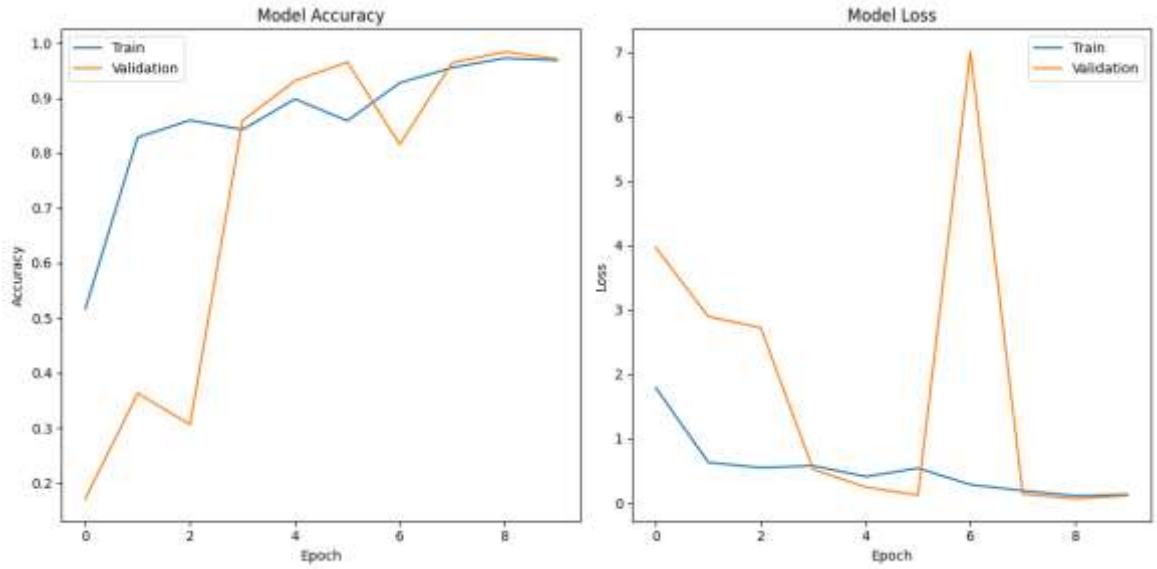


Fig. 11: ResNet50 model metrics

Fig. 12: ResNet50 model output

8.3 YOLOv8 Model METRICS:

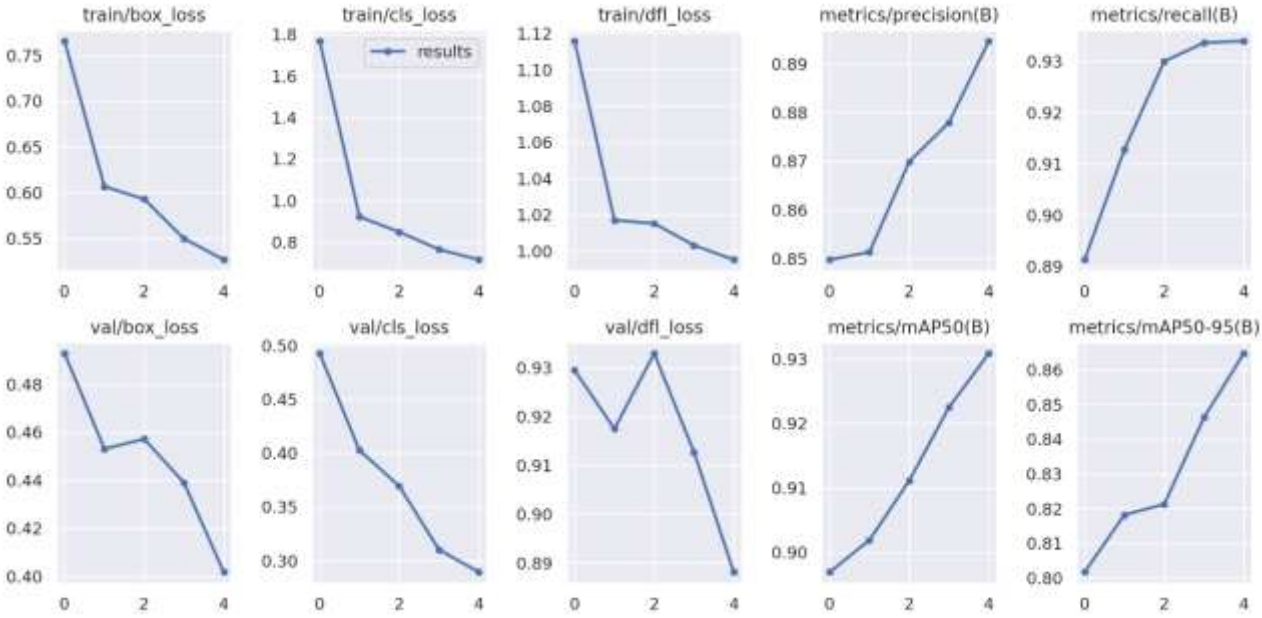


Fig. 13: YOLOv8 model metrics

Fig. 14: YOLOv8 model output

RESULTS COMPARISION

Approach Used	Accuracy	Precision	Recall	F1-score
Our Work:				
CNN	0.98	0.97	0.95	0.98
ResNet50	0.93	0.91	0.90	0.90
YOLOv8	0.94	0.91	0.89	0.89
Previous Work:				
YOLOv5	0.88	0.82	0.79	-
VSSA-net	0.94	0.78	-	-
Alexnet	0.92	0.90	-	-
Faster RCNN	0.92	0.93	0.94	0.93

Table 3: Results Comparison

- The CNN model trained on GTSRB data achieved a remarkable accuracy of 98%, showcasing its proficiency in recognizing traffic signs. With a weighted precision of 97% and a weighted F1-score of 98%, the model demonstrates high precision and overall performance. Despite the high precision, the weighted recall of 95% indicates a slight compromise in identifying all relevant instances.

- The ResNet50 model, trained on GTSRB data, achieved a commendable test accuracy of 93% with a low loss of 0.43, indicating robust performance in traffic sign recognition. Its precision of 91%, recall of 90%, and F1-score of 90% underscore its balanced ability to accurately classify signs while minimizing false positives and negatives. Despite being trained over 100 epochs, the model demonstrates efficient learning and generalization, affirming its reliability for real-world traffic sign recognition tasks.

- The YOLOv8 model trained on GTSRB data yielded strong results, with an accuracy of 94% and precision of 91%, indicating its capability to effectively detect and classify traffic signs. Despite a slightly lower recall and F1-score of 89%, the model still demonstrates a balanced performance in identifying relevant instances while minimizing false positives. These metrics suggest the model's proficiency in real-world traffic sign detection tasks, making it a reliable tool for applications requiring high accuracy and precision.

CHAPTER - 9
CONCLUSION & FUTURE SCOPE

In conclusion, this project addresses the critical challenges faced by autonomous vehicles in accurately detecting traffic signs, especially in adverse weather conditions. By building upon insights from prior research and acknowledging limitations associated with small datasets and varied weather scenarios, our proposed solution employs a comprehensive approach. Through the integration of benchmark datasets, innovative data augmentation techniques, and the utilization of state-of-the-art deep learning models such Convolution Neural Network (CNN), and ResNet50, we aim to significantly enhance the robustness and accuracy of traffic sign detection. This endeavour not only contributes to overcoming existing limitations but also paves the way for more reliable and safer autonomous navigation, ultimately advancing the field of autonomous driving technology.

The future scope of this project extends to various avenues, including the integration of real-time data streams for dynamic adaptation to changing weather conditions, refinement of the deep learning models for improved accuracy and efficiency, and exploration of multi-sensor fusion techniques to further enhancement the robustness of the traffic sign detection system. Additionally, the insights gained from this project can be applied to other domains requiring resilient perception systems, such as surveillance, robotics, and augmented reality, thereby contributing to the broader advancement of computer vision technology.

CHAPTER - 10

REFERENCES

[1] Rajesh Kannan Megalingam, Kondareddy Thanigundala, Sreevatsava Reddy Musani, Hemanth Nidamanuru, Lokesh Gadde, Indian traffic sign detection and recognition using deep learning, International Journal of Transportation Science and Technology, Volume 12, Issue 3, 2023.

[2] Qu S., Yang, X., Zhou, H. et al. Improved YOLOv5-based for small traffic sign detection under complex weather. Sci Rep 13, 16219 (2023).

[3] Wang Rui, Zhao He, Xu Zhengwei, Ding Yaming, Li Guowei, Zhang Yuxin, Li Hua, Real-time vehicle target detection in inclement weather conditions based on YOLOv4, Volume-17, 2023.

[4] Liu, W.; Ren, G.; Yu, R.; Guo, S.; Zhu, J.; Zhang, L. Image-adaptive YOLO for object detection in adverse weather conditions 2022.

[5] Zhang C, Yue X, Wang R, Li N, Ding Y. Study on traffic sign recognition by optimized lenet-5 algorithm. Int J Pattern Recog Artif Intell. 2020.

[6] Tai S-K, Dewi C, Chen R-C, Liu Y-T, Jiang X, Yu H. Deep learning for traffic sign recognition based on spatial pyramid pooling with scale analysis. Appl Sci. 2020.

[7] Rajendran, S.P.; Shine, L.; Pradeep, R.; Vijayaraghavan, S. Real-time traffic sign recognition using YOLOv3 based detector. 2019.

[8] Cire san D, Meier U, Masci J, Schmidhuber J (2012) Multi-column deep neural network for traffic sign classification. Neural networks. In: The international joint conference on neural network, IDSIA-USI-SUPSI Galleria, vol 2

[9] Yuan Y, Xiong Z, Wang Q. Vssa-net: vertical spatial sequence attention network for traffic sign detection. IEEE Trans Image Process. 2019;28(7):3423–3434. doi: 10.1109/TIP.2019.2896952.

[10] Huang S-C, Lin H-Y, Chang C-C (2017) An in-car camera system for traffic sign detection and recognition. In: 2017 Joint 17th world congress of international fuzzy systems association and 9th international conference on soft computing and intelligent systems (IFSA-SCIS). IEEE, pp 1–6

[11] Bi Z, Yu L, Gao H, Zhou P, Yao H (2020) Improved vgg model-based efficient traffic sign recognition for safe driving in 5g scenarios. Int J Mach Learn Cybern:1–12

[12] Han C, Gao G, Zhang Y. Real-time small traffic sign detection with revised faster-rcnn. *Multimed Tools Appl.* 2019;78(10):13263–13278. doi: 10.1007/s11042-018-6428-0.

[13] Zhang C, Yue X, Wang R, Li N, Ding Y. Study on traffic sign recognition by optimized lenet-5 algorithm. *Int J Pattern Recog Artif Intell.* 2020;34(01):2055003. doi: 10.1142/S0218001420550034.

[14] Qian R, Zhang B, Yue Y, Wang Z, Coenen F (2015) Robust chinese traffic sign detection and recognition with deep convolutional neural network. In: 2015 11th International conference on natural computation (ICNC). IEEE, pp 791–796

[15] Liu C, Tao Y, Liang J, Li K, Chen Y (2018) Object detection based on yolo network. In: 2018 IEEE 4th Information technology and mechatronics engineering conference (ITOEC). IEEE, pp 799–803

[16] Mehta S, Paunwala C, Vaidya B (2019) Cnn based traffic sign classification using adam optimizer. In: 2019 International conference on intelligent computing and control systems (ICCS). IEEE, pp 1293–1298

[17] Isa ISBM, Choy JY, Shaari NLABM. Real-time traffic sign detection and recognition using raspberry pi. *Int J Electr Comput Eng.* 2022;12(1):331.

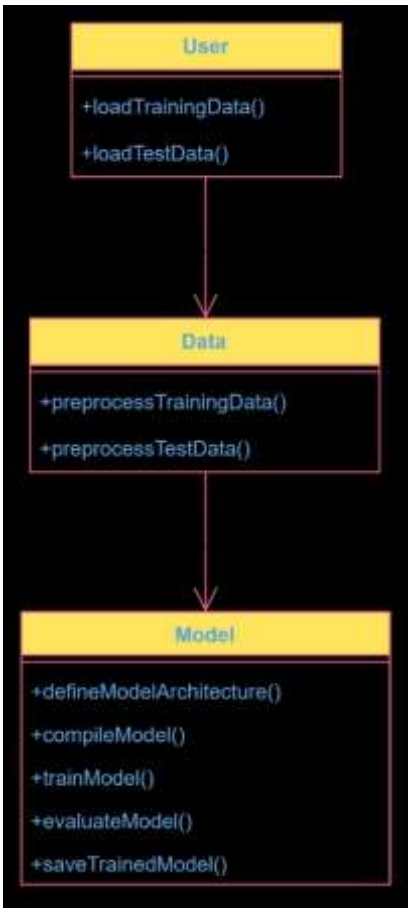
APPENDIX – I
DESIGN DOCUMENTS

UML DIAGRAMS

UML aims to establish a consistent method for illustrating system designs, akin to blueprints used in other engineering disciplines. It serves as a universal modeling language within software engineering, offering a standardized approach to visualizing system designs.

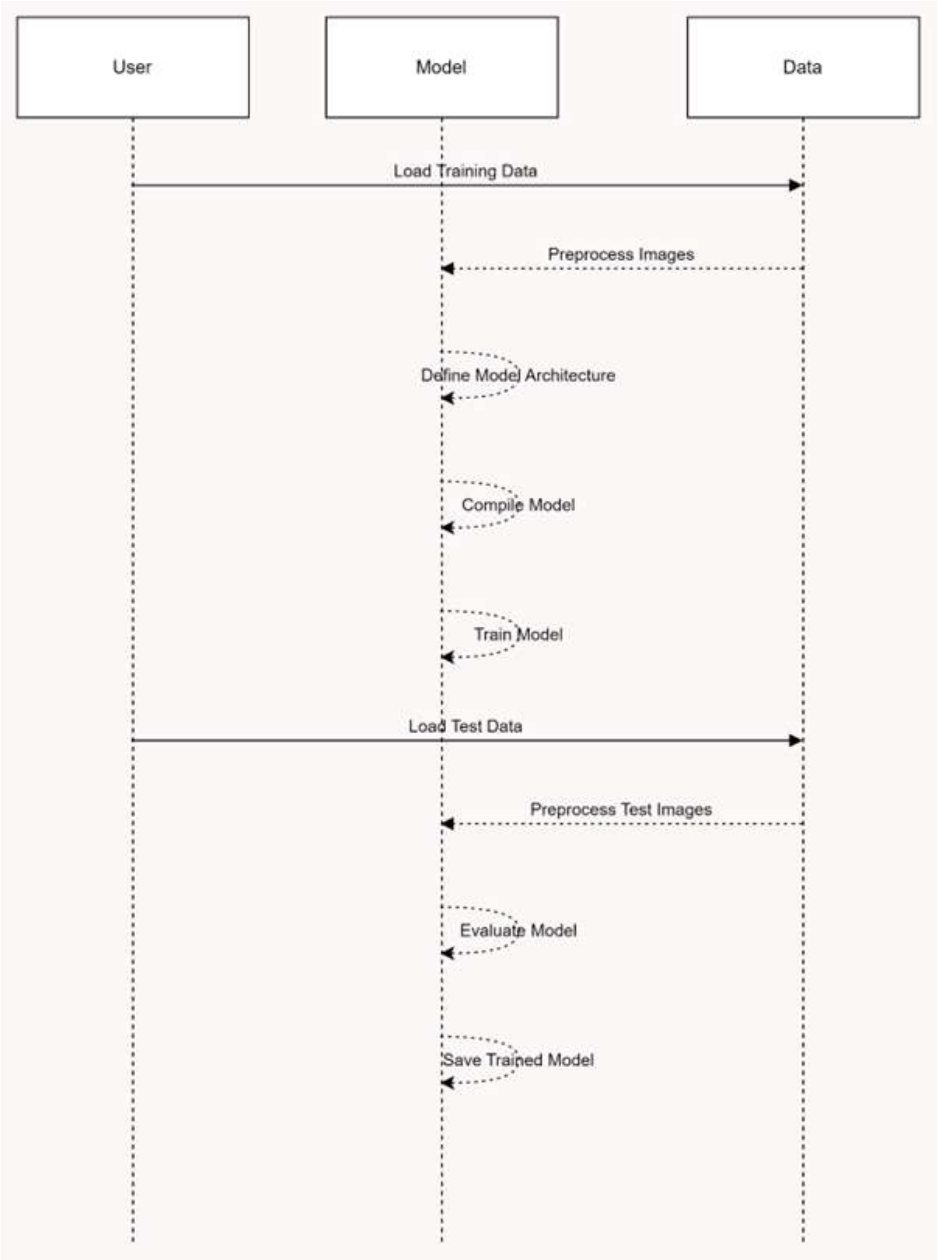
A.I.1 CLASS DIAGRAM

A class diagram is a visual representation of the structure and relationships among classes in a system. It depicts the static structure of the system, including classes, attributes, operations, and their associations. Class diagrams are crucial in software design for modeling the entities and their relationships in a system, aiding in communication among stakeholders, guiding the implementation process, and facilitating the understanding of system architecture. Key properties include showing the hierarchy of classes, associations between classes, attributes, and methods, and providing a blueprint for system design and development.



A.I.2 SEQUENCE DIAGRAM

A sequence diagram is a visual representation of interactions between objects or components in a system, showing the sequence of messages exchanged over time. It helps in understanding the flow of control and data during execution. Sequence diagrams are essential in software design for illustrating the dynamic behaviour of systems, aiding in communication among stakeholders, identifying potential issues in system interactions, and facilitating the design of efficient and reliable systems. Key properties include depicting the chronological order of messages, showing concurrent activities, and highlighting the dependencies between components.



APPENDIX – II

SAMPLE CODE

A.II.1 CNN Code:

Importing Required Libraries

```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from PIL import Image
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score
np.random.seed(42)

from matplotlib import style
style.use('fivethirtyeight')
```

Assigning Path for Dataset

```
data_dir = '../input/gtsrb-german-traffic-sign'
train_path = '../input/gtsrb-german-traffic-sign/Train'
test_path = '../input/gtsrb-german-traffic-sign/'

# Resizing the images to 30x30x3
IMG_HEIGHT = 30
IMG_WIDTH = 30
channels = 3
```

Finding Total Classes

```
NUM_CATEGORIES = len(os.listdir(train_path))
NUM_CATEGORIES
```

```
43
```

```
# Label Overview
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
```

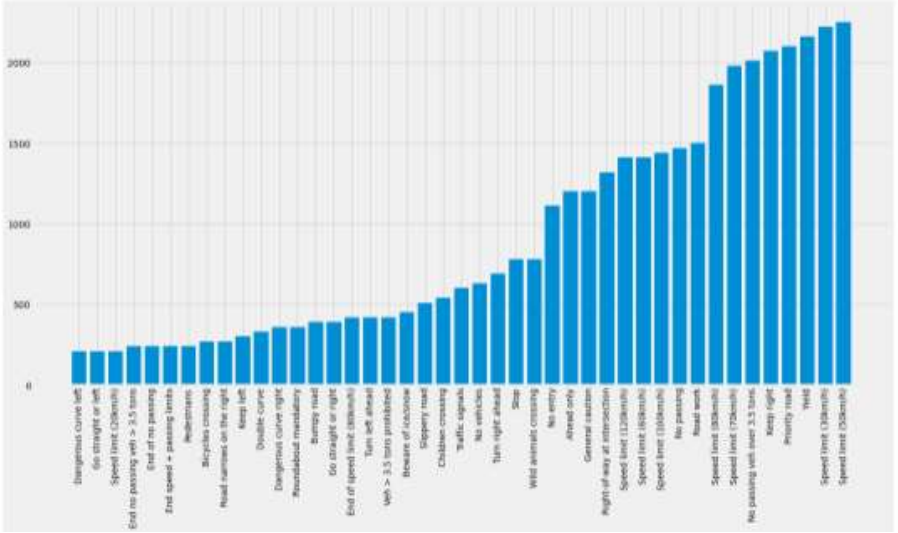
```
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Veh > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
```

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

Visualizing The Dataset

```
folders = os.listdir(train_path)

train number = []
class_num = []

for folder in folders:
    train files = os.listdir(train path + '/' + folder)
    train_number.append(len(train_files))
    class_num.append(classes[int(folder)])
```

```
# Sorting the dataset on the basis of number of images in each class
zipped_lists = zip(train_number, class_num)
sorted_pairs = sorted(zipped_lists)
tuples = zip(*sorted_pairs)
train_number, class_num = [ list(tuple) for tuple in  tuples]

# Plotting the number of images in each class
plt.figure(figsize=(21,10))
plt.bar(class_num, train_number)
plt.xticks(class_num, rotation='vertical')
plt.show()
```



```
# Visualizing 25 random images from test data
import random
from matplotlib.image import imread

test = pd.read_csv(data_dir + '/Test.csv')
imgs = test["Path"].values

plt.figure(figsize=(25,25))

for i in range(1,26):
    plt.subplot(5,5,i)
    random_img_path = data_dir + '/' + random.choice(imgs)
    rand_img = imread(random_img_path)
    plt.imshow(rand_img)
    plt.grid(b=None)
```

```
    plt.xlabel(rand_img.shape[1], fontsize = 20)#width of image
    plt.ylabel(rand_img.shape[0], fontsize = 20)#height of image
```

Collecting the Training Data

```
image_data = []
image_labels = []

for i in range(NUM_CATEGORIES):
    path = data_dir + '/Train/' + str(i)
    images = os.listdir(path)

    for img in images:
        try:
```

```
            image = cv2.imread(path + '/' + img)
            image_fromarray = Image.fromarray(image, 'RGB')
            resize_image = image_fromarray.resize((IMG_HEIGHT,
IMG_WIDTH))
            image data.append(np.array(resize_image))
            image_labels.append(i)
        except:
            print("Error in " + img)

# Changing the list to numpy array
image data = np.array(image data)
image_labels = np.array(image_labels)

print(image_data.shape, image_labels.shape)
```

```
(39209, 30, 30, 3) (39209,)
```

Shuffling the training data

```
shuffle_indexes = np.arange(image_data.shape[0])
np.random.shuffle(shuffle indexes)
image_data = image_data[shuffle_indexes]
image_labels = image_labels[shuffle_indexes]
```

Splitting the data into train and validation set

```
X_train, X_val, y_train, y_val = train_test_split(image_data,
image_labels, test_size=0.3, random_state=42, shuffle=True)

X train = X train/255
X_val = X_val/255

print("X_train.shape", X_train.shape)
print("X_valid.shape", X_val.shape)
print("y_train.shape", y_train.shape)
print("y_valid.shape", y_val.shape)
```

```
X_train.shape (27446, 30, 30, 3)
X_valid.shape (11763, 30, 30, 3)
y train.shape (27446,)
y_valid.shape (11763,)
```

One hot encoding the labels

```
y_train = keras.utils.to_categorical(y_train, NUM_CATEGORIES)
y_val = keras.utils.to_categorical(y_val, NUM_CATEGORIES)

print(y train.shape)
print(y_val.shape)
```

```
(27446, 43)
(11763, 43)
```

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=16, kernel_size=(3,3),
activation='relu', input_shape=(IMG_HEIGHT,IMG_WIDTH,channels)),
    keras.layers.Conv2D(filters=32, kernel_size=(3,3),
activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Conv2D(filters=64, kernel_size=(3,3),
activation='relu'),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3),
activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Flatten(),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate=0.5),

    keras.layers.Dense(43, activation='softmax')
])

lr = 0.001
epochs = 30

opt = Adam(lr=lr, decay=lr / (epochs * 0.5))
model.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
```

Augmenting the data and training the model

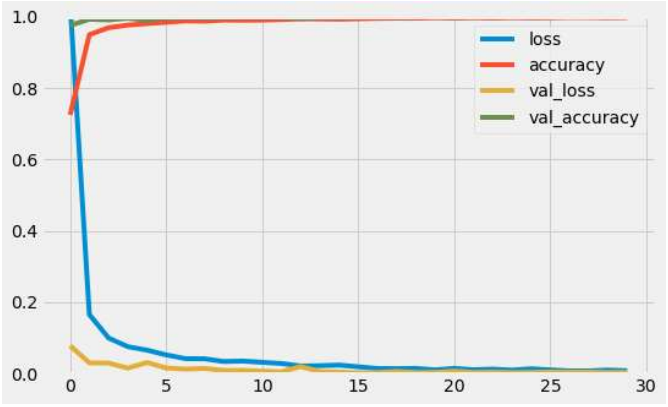
```
aug = ImageDataGenerator(
    rotation_range=10,
    zoom range=0.15,
    width_shift_range=0.1,
    height shift range=0.1,
    shear_range=0.15,
    horizontal flip=False,
    vertical_flip=False,
    fill_mode="nearest")

history = model.fit(aug.flow(X train, y train, batch_size=32),
epochs=epochs, validation_data=(X_val, y_val))
```

```
Epoch 1/30
858/858 [==============================] - 17s 19ms/step - loss:
1.0365 - accuracy: 0.7248 - val_loss: 0.0784 - val_accuracy: 0.9751
Epoch 2/30
858/858 [==============================] - 16s 18ms/step - loss:
0.1657 - accuracy: 0.9493 - val_loss: 0.0304 - val_accuracy: 0.9924
Epoch 3/30
858/858 [==============================] - 16s 19ms/step - loss:
0.1001 - accuracy: 0.9687 - val_loss: 0.0298 - val_accuracy: 0.9906
Epoch 4/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0758 - accuracy: 0.9762 - val_loss: 0.0158 - val_accuracy: 0.9953
Epoch 5/30
858/858 [==============================] - 17s 19ms/step - loss:
0.0661 - accuracy: 0.9801 - val_loss: 0.0315 - val_accuracy: 0.9897
Epoch 6/30
858/858 [==============================] - 17s 20ms/step - loss:
0.0528 - accuracy: 0.9840 - val_loss: 0.0161 - val_accuracy: 0.9957
```

```
0.0145 - accuracy: 0.9949 - val_loss: 0.0033 - val_accuracy: 0.9989
Epoch 18/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0144 - accuracy: 0.9955 - val_loss: 0.0064 - val_accuracy: 0.9980
Epoch 19/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0150 - accuracy: 0.9953 - val_loss: 0.0033 - val_accuracy: 0.9991
Epoch 20/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0108 - accuracy: 0.9965 - val_loss: 0.0037 - val_accuracy: 0.9989
Epoch 21/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0151 - accuracy: 0.9956 - val_loss: 0.0071 - val_accuracy: 0.9982
Epoch 22/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0112 - accuracy: 0.9967 - val_loss: 0.0036 - val_accuracy: 0.9986
Epoch 23/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0130 - accuracy: 0.9958 - val_loss: 0.0043 - val_accuracy: 0.9988
Epoch 24/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0102 - accuracy: 0.9969 - val_loss: 0.0031 - val_accuracy: 0.9988
Epoch 25/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0140 - accuracy: 0.9961 - val_loss: 0.0038 - val_accuracy: 0.9990
Epoch 26/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0108 - accuracy: 0.9967 - val_loss: 0.0048 - val_accuracy: 0.9986
Epoch 27/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0079 - accuracy: 0.9977 - val_loss: 0.0045 - val_accuracy: 0.9988
Epoch 28/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0080 - accuracy: 0.9974 - val_loss: 0.0045 - val_accuracy: 0.9988
Epoch 29/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0103 - accuracy: 0.9966 - val_loss: 0.0036 - val_accuracy: 0.9991
Epoch 30/30
858/858 [==============================] - 16s 19ms/step - loss:
0.0084 - accuracy: 0.9973 - val_loss: 0.0040 - val_accuracy: 0.9989
```

Evaluating the model

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



Loading the test data and running the predictions

```
test = pd.read_csv(data_dir + '/Test.csv')

labels = test["ClassId"].values
imgs = test["Path"].values

data =[]

for img in imgs:
    try:
        image = cv2.imread(data dir + '/' +img)
        image_fromarray = Image.fromarray(image, 'RGB')
        resize image = image fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
        data.append(np.array(resize_image))
    except:
        print("Error in " + img)
X_test = np.array(data)
X_test = X_test/255

pred = model.predict_classes(X_test)

#Accuracy with the test data
print('Test Data accuracy: ',accuracy_score(labels, pred)*100)

Test Data accuracy:  98.86777513855898
```

Predictions on Test Data

```
plt.figure(figsize = (25, 25))

start index = 0
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    prediction = pred[start_index + i]
    actual = labels[start_index + i]
    col = 'g'
    if prediction != actual:
        col = 'r'
    plt.xlabel('Actual={} || Pred={}'.format(actual, prediction),
color = col)
    plt.imshow(X_test[start_index + i])
plt.show()
```

Classification report

```
from sklearn.metrics import classification_report

print(classification_report(labels, pred))
```

	precision	recall	f1-score	support
0	0.97	1.00	0.98	60
1	0.99	1.00	0.99	720
2	0.98	1.00	0.99	750
3	0.99	0.98	0.98	450
4	1.00	1.00	1.00	660
5	0.99	0.99	0.99	630
6	1.00	0.97	0.98	150
7	1.00	1.00	1.00	450
8	1.00	0.99	0.99	450
9	1.00	1.00	1.00	480
10	1.00	1.00	1.00	660
11	0.99	0.98	0.99	420
12	0.99	0.99	0.99	690
13	1.00	1.00	1.00	720
14	1.00	1.00	1.00	270
15	0.99	1.00	0.99	210
16	1.00	1.00	1.00	150
17	1.00	0.99	0.99	360
18	1.00	0.95	0.97	390
19	1.00	1.00	1.00	60
20	0.98	1.00	0.99	90
21	0.81	1.00	0.90	90
22	1.00	0.99	1.00	120
23	0.94	1.00	0.97	150
24	0.99	0.96	0.97	90
25	0.99	0.99	0.99	480
26	0.97	0.99	0.98	180
27	0.97	0.63	0.77	60
28	0.99	0.95	0.97	150
29	0.79	1.00	0.88	90
30	0.95	0.91	0.93	150
31	0.99	1.00	0.99	270
32	1.00	1.00	1.00	60
33	1.00	1.00	1.00	210
34	1.00	1.00	1.00	120
35	1.00	1.00	1.00	390
36	0.98	0.99	0.98	120
37	0.97	0.98	0.98	60
38	1.00	1.00	1.00	690
39	0.99	0.97	0.98	90
40	1.00	0.96	0.98	90
41	1.00	0.98	0.99	60
42	0.99	1.00	0.99	90
accuracy			0.99	12630
macro avg	0.98	0.98	0.98	12630
weighted avg	0.99	0.99	0.99	12630

APPENDIX – III

PLAGIARISM REPORT

ABSTRACT

Autonomous vehicles heavily rely on accurate and timely detection of traffic signs for safe navigation. In the process of traffic sign detection, the size of the objects and weather conditions exhibit significant variability, which can influence the detection accuracy. This project aims to enhance the robustness in detecting the traffic signs in challenging weather conditions through the implementation of the deep learning techniques.

Adverse weather conditions pose significant challenges for autonomous vehicles (AVs), particularly in rain, snow, and similar scenarios. To address these challenges, we introduce a deep learning system designed for the enhancement of the detection of traffic signs, providing a clear understanding for Autonomous vehicles. Our proposed system aims to optimize traffic sign board detection, enabling reliable performance in all weather conditions. Through the utilization of deep learning techniques, we seek to unlock the potential for all-weather AV navigation, contributing to safer and more robust autonomous driving experiences.

LIST OF TABLES

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

In the domain of autonomous vehicles, the ability to accurately perceive and interpret traffic signs under varying environmental conditions is essential for ensuring safe navigation. However, traditional computer vision algorithms often struggle in challenging weather conditions or low-light situations, hindering reliable traffic sign detection and compromising the safety of autonomous driving systems. To address this challenge, our project, titled "Enhancing Traffic Sign Detection for Autonomous Vehicles in Challenging Weather Conditions through Deep Learning," aims to develop an advanced solution. Our approach involves utilizing deep learning techniques to significantly improve traffic sign detection performance.

The rise of autonomous vehicles marks a significant transformation in transportation systems worldwide. An essential aspect of their safe operation is their ability to detect and interpret traffic signs accurately in real-time. However, challenging weather conditions pose a significant obstacle to this objective. Traditional computer vision algorithms often struggle to maintain reliable performance when faced with reduced visibility, changes in lighting, and occlusions caused by variability in weather conditions. Consequently, ensuring consistent traffic sign detection under such challenging circumstances remains a crucial hurdle for the widespread adoption of autonomous driving systems.

Our project aims to develop an adaptive vision system capable of reliably detecting traffic signs across diverse weather conditions by leveraging deep learning techniques. Through convolutional neural networks (CNNs) and recurrent neural networks (RNNs), we aim to extract detailed features from traffic sign images, enabling the system to generalize effectively across different weather conditions and lighting scenarios.

The significance of our project extends beyond autonomous vehicles. By enhancing the perceptual capabilities of automated driving systems, our solution has the potential to improve safety, efficiency, and reliability in transportation networks globally. Furthermore, the methodologies and techniques developed through this project offer valuable insights into addressing challenges related to adverse environmental conditions in various applications beyond autonomous driving.

In conclusion, our project represents an innovative effort to tackle the critical challenge of traffic sign detection in adverse weather conditions using different deep learning techniques. Through this endeavor, we aim to contribute significantly to the advancement of autonomous driving technology, ultimately paving the way for safer and more efficient transportation systems in the future.

CHAPTER - 2

LITERATURE SURVEY

- [2] In the work of Qu and Yang, to develop a YOLOv5 model, the researchers applied the CCTSDB 2021 dataset in the training process. However, their model achieved a relatively low accuracy level of 88.1%, which complicates its application in practice. Additionally, the selected dataset does not represent the standard benchmark for the target task.

- In the work, "Multi - column deep neural network for traffic sign classification", Cire et al. developed and explored the performance of the MCDNN model using the GTSRB dataset. With the impressive accuracy of 99.46%, the model's performance was significantly high. Nevertheless, it underperformed when provided with the blurred images, and it is necessary to investigate the implications of blurry visual data for the performance of the original and similar models. In the process, the existing solutions cannot be considered as the optimal choice due to the above-mentioned factor and it is required to conduct new research and create a new efficient and smart system.

- In Ref. [9], the authors introduced the VSSA-net model and evaluated its performance using the VOC 2007 dataset. According to the results, the proposed approach could achieve an accuracy of 94.6%. However, it was also noted that the model developed by Yuan et al. demonstrated low performance on large datasets. Thus, further research should be conducted to improve the scalability and generalization of the proposed approach for small datasets.

- Shu-Chun Huang has experimented the AlexNet architecture on the GTSDB dataset and obtained a good result of 92.63% in terms of accuracy. The drawback of this approach is that it failed to well distinguish actual traffic signs from the background; this suggests that the feature extraction and discrimination mechanisms must be upgraded to improve the accuracy and reliability of the used solution for real-world traffic sign recognition.

- [11] An upgraded version of the VGG model was presented by Zhongqin Bi and its performance on the GTSRB dataset evaluated with an impressive accuracy of 98.47%. But in the study, they never mentioned anything about how it would work under harsh weather conditions. In other words, this can be a limitation that shows why further research is necessary so as to improve the toughness of the model and make it suitable for different environmental settings including those that are unfavorable.

- Cen Han applied the Faster R-CNN model and assessed its effectiveness on his personal data set, resulting in 92.00% accuracy that was quite good. However, one weakness of this method is the increased detection time as against other models which indicate a balance between efficiency and accuracy. This highlights the importance of further optimization to enhance the speed of the model while keeping its detection performance for real-time applications helpful [12].

- Chuanwei Zhang employed the LeNet-5 architecture and assessed its performance on the GTSDB dataset, achieving a respectable accuracy of 94% [13]. However, a notable limitation of this implementation was its slower response time, indicating a potential trade-off between accuracy and speed. This highlights the importance of optimizing the model's architecture or utilizing hardware acceleration techniques to improve its real-time performance for time-sensitive applications.

- Zhao Wang's work involves the usage of a CNN model, which was evaluated on one database, the GTSRB database, with an accuracy of 95.0%. The technique did not take into account the tough environmental circumstances found in real-world situations reported in [14]. As a result of this neglect, the model's flexibility and generalizability may suffer. As a result, future research should focus on addressing this issue in order to improve the practical use of models under a variety of environmental conditions.

- [15] Chengji Liu utilized the YOLOc (You Only Look Once) model and evaluated its performance on the GTSDB dataset, achieving a commendable accuracy of 94%. However, a drawback of this implementation was its slower response time, indicating a compromise between accuracy and speed. This highlights the need for further optimization to enhance the model's efficiency without sacrificing its detection accuracy, particularly for real-time applications where swift processing is crucial.

- Smit Mehta implemented a CNN model and evaluated its performance on the BTSD dataset, achieving a high accuracy of 98.26%. However, the model exhibited poor performance when presented with blurred images, indicating a limitation in its ability to effectively handle such scenarios. This underscores the importance of enhancing the model's robustness to handle diverse and challenging image conditions, including blurred images, for improved real-world applicability [16].

- [17] Choy Ja Yeong utilized a Raspberry Pi-based system for traffic sign recognition and evaluated its performance on the GTSRB dataset, achieving an accuracy of 90.00%. However, one notable drawback of this implementation was the greater detection time compared to other systems or models. This suggests a trade-off between accuracy and processing speed on the Raspberry Pi platform, highlighting the need for optimization strategies to improve real-time performance while maintaining satisfactory accuracy for practical applications.

CHAPTER – 3

OBSERVATIONS FROM LITERATURE SURVEY

- Within our in-depth exploration of the present research for the project titled "Enhancing Traffic Sign Detection in Harsh Weather Conditions for Autonomous Vehicles Using Deep Learning", we have managed to identify some crucial insights that stemmed from past studies.

- Notably, research papers have employed various deep learning models such as R-CNN, customized CNN, older versions of YOLO (v2, v3, v5) architectures to tackle the challenges of traffic sign detection. While some studies have reported promising results, there's a consensus on the difficulties posed by adverse weather conditions. A subset of these investigations has shed light on the limitations associated with small datasets, leading to decreased accuracy levels.

- These findings emphasize the critical need to expand datasets and explore more sophisticated deep learning architectures. This literature survey lays the groundwork for our project, highlighting the importance of addressing existing challenges through innovative approaches and advancements in the field.

- We've noted a common concern: the demanding nature of object detection in autonomous driving scenarios. While our literature review has uncovered valuable insights, we acknowledge a prevailing gap in the consideration of occlusion, a critical factor in real-world applications.

Existing systems also mentioned that the framework can be improved through data augmentation and object persistence.

Key points to be considered for further implementation:

- Using large and benchmark dataset(s) for good accuracy.
- Considering the situation of Object Occlusion.
- Improving the framework with data augmentation and object persistence.

CHAPTER - 4

PROBLEM STATEMENT

In adverse weather conditions, conventional computer vision often struggle to accurately detect traffic signs due factors like poor visibility, altered lighting, and obstructions such as raindrops, snowflakes, or fog. This deficiency poses a significant safety risk for autonomous vehicles, as reliable traffic sign detection is crucial for making critical driving decisions promptly and effectively. Existing solutions primarily rely on handcrafted features and rule-based algorithms, which lack the adaptability and resilience needed to cope with the diverse challenges posed by adverse weather conditions.

To address this critical limitation, our project seeks to pioneer a transformative approach by leveraging advanced deep learning algorithms. By harnessing the power of deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), we aim to develop a robust and adaptable traffic sign detection system capable of operating effectively across a spectrum of challenging weather conditions. Through this endeavor, we aspire to significantly enhance the safety and reliability of autonomous vehicles, ushering in a new era of intelligent transportation systems

CHAPTER - 5

METHODOLOGY

5.1 PROPOSED SOLUTION

In response to the identified drawbacks in previous research, our proposed project aims to address these challenges comprehensively.

- We intend to mitigate limitations associated with small datasets by incorporating benchmark datasets, fostering increased diversity and scale. Employing innovative techniques like data augmentation will further augment our dataset, enhancing the model's ability to generalize and improving overall accuracy.

- To specifically tackle the adverse weather conditions that pose challenges to traffic sign detection, our approach involves considering a diverse range of weather scenarios during both training and testing phases.

- In addition, we plan to leverage advanced deep learning models, including Convolution Neural Network (CNN), YOLOv8, ResNet50, and others, to elevate the sophistication and efficiency of our traffic sign detection system.

Through these strategic measures, our project aims to significantly enhance the robustness and performance of traffic sign detection in adverse weather conditions for autonomous vehicles, contributing to the advancement of safety and reliability in autonomous navigation.

5.2 PROPOSED IMPLEMENTATION

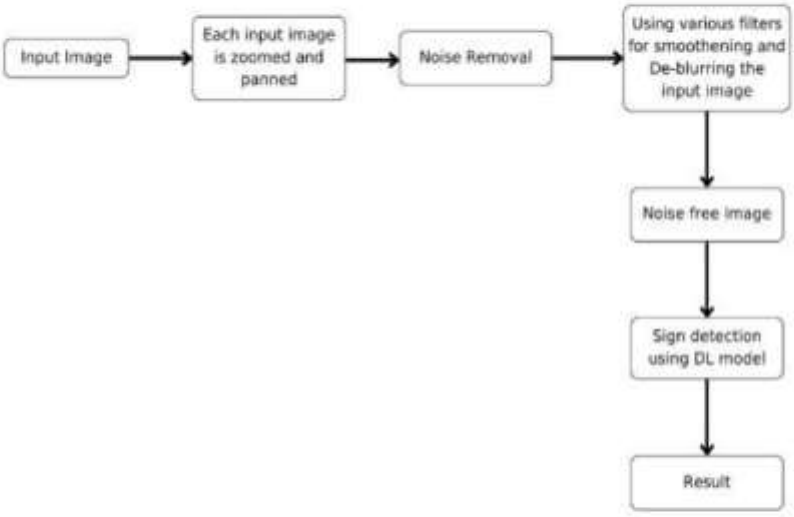


Fig. 1: Proposed Implementation

- **Video Capture:** HD cameras mounted on autonomous vehicles capture video footage, sampled at specific frame rate. Environmental conditions enhance image quality to provide clear input for subsequent processing stages.

- **Preprocessing:** Captured video undergoes preprocessing to enhance image quality and prepare for analysis. Different techniques like Median filtering, Lucy Richardson filter and Histogram equalization etc., are used for removing noise from images.

- **Traffic Sign Recognition & Detection:** Recognition & Detection of traffic signs must be quick and efficient to ensure proper functioning of the system so, algorithms like CNN, YOLOv8, ResNet50 etc., are employed for real-time detection of traffic signs.

CHAPTER - 6

DATASET OVERVIEW

- The German Traffic Sign Recognition Benchmark (GTSRB) dataset is a widely used dataset in the field of computer vision, especially in the field of traffic sign recognition. It is designed to develop and evaluate algorithms for the task of classifying traffic signs commonly found on roads.
- The German Traffic Sign Recognition Test (GTSRB) includes 43 types of traffic signs, divided into 39,209 training images and 12,630 test images.
- Images feature different lighting conditions and rich backgrounds.

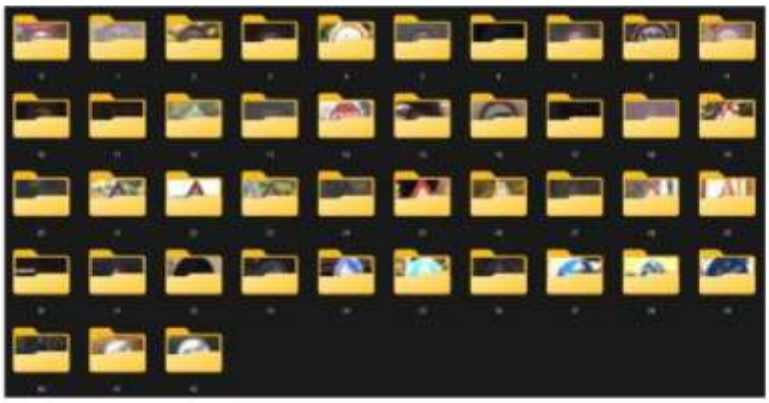


Fig. 2: Dataset Overview

- Different classes of GTSRB dataset:

1. Speed limit (20 km/h)
2. Speed limit (30 km/h)
3. Speed limit (50 km/h)
4. Speed limit (60 km/h)
5. Speed limit (70 km/h)
6. Speed limit (80 km/h)
7. End of speed limit (80 km/h)
8. Speed limit (100 km/h)
9. Speed limit (120 km/h)
10. No passing
11. No passing for vehicles over 3.5 metric tons
12. Right-of-way at the next intersection

13. Priority road
14. Yield
15. Stop
16. No vehicles
17. Vehicles over 3.5 metric tons prohibited
18. No entry
19. General caution
20. Dangerous curve to the left
21. Dangerous curve to the 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 of all speed and 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 of no passing by vehicles over 3.5 metric tons

- Number of samples per class:

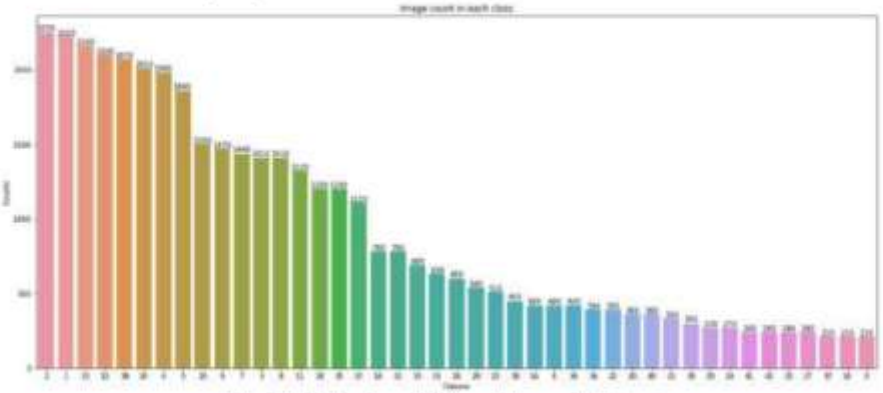


Fig. 3: Distribution of different classes in dataset

12

- Sample images from GTSRB dataset:

Fig. 4: Sample images of Dataset

Reference: Dataset Link

13

CHAPTER - 7

IMPLEMENTATION

- As a part of our research project, we have implemented various Machine Learning and Deep Learning models to get the most accurate one.
- We have trained the models with a training data of 39,209 images and had done the testing and validation on 12,630 test images which in whole contain 43 different classes of traffic signs.
- The implemented models are:
  1. Convolution Neural Network (CNN)
  2. Residual Network (ResNet50)

7.1 CONVOLUTION NEURAL NETWORK (CNN)

CNN Architecture

- The term "CNN" stands for Convolutional Neural Network. It's a type of artificial neural network, particularly suited for processing structured grid-like data, such as images.
- Different layers of CNN architecture used in this model are:
  - Input Layer: Accepts images with dimensions (IMG_HEIGHT, IMG_WIDTH, channels), where IMG_HEIGHT and IMG_WIDTH represent the image dimensions, and 'channels' represents the number of color channels (e.g., 3 for RGB).
  - Convolutional Layers: Consist of two pairs of Conv2D layers. Each pair has two Conv2D layers with increasing numbers of filters (16 in the first pair, 32 and 64 in the second pair) and ReLU activation functions.
  - MaxPooling Layers: Follow each pair of convolutional layers to downsample the feature maps using max-pooling with a pool size of (2, 2).

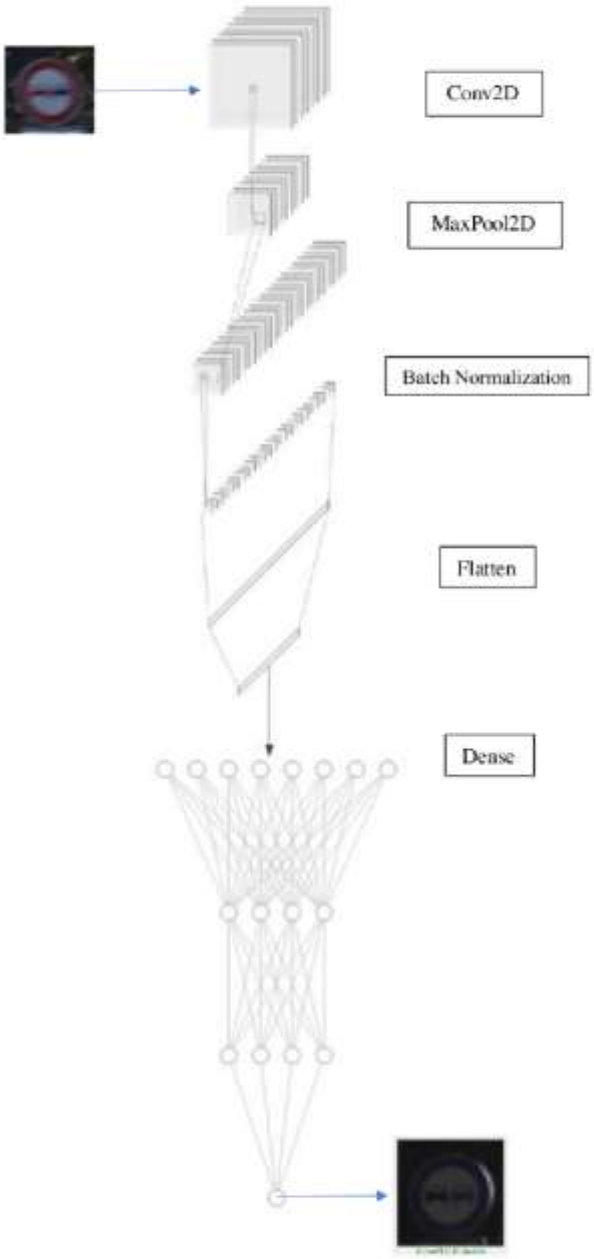


Fig. 5: CNN Architecture

15

- Batch Normalization: These layers are added after each pair of convolutional layers to normalize activation, improving stability and training speed.
- Flatten Layer: Convert 2D feature maps to 1D feature vectors, which will be fed into fully connected layers.
- Fully Connected Layers: Includes a dense layer with 512 units and ReLU activation function. Batch normalization is typically applied before the dropout layer with a dropout rate of 0.5, to reduce overfitting and improve model generalization.
- Output Layer: Consists of a dense layer with 43 units (equal to the number of layers) and a softmax activation function, which generates a prediction probability for each layer.

Overall, this CNN architecture is designed for multi-class classification tasks, where the goal is to classify the input image into one of 43 classes. The model uses convolutional layers for feature extraction, max pooling layers for spatial reduction, and fully connected layers for classification. Normalization and batch cancellation are integrated to improve training stability and reduce overfitting.

7.2 RESIDUAL NETWORK (ResNet50)

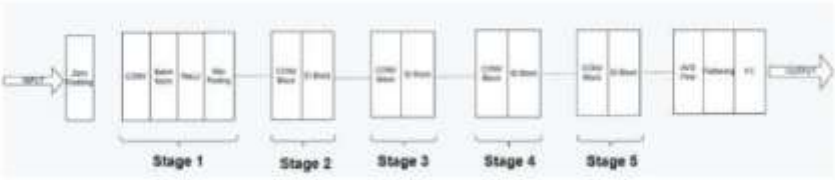


Fig. 6: ResNet50 Architecture

ResNet50 Architecture

- ResNet50 is a convolutional neural network (CNN) architecture that was introduced by Kaiming He et al. in their paper titled "Deep Residual Learning for Image Recognition" in 2015. It is part of the ResNet50 (Residual Network 50) family of models, which are renowned for their effectiveness in training very deep neural networks.
- Here are the different layers of the ResNet50 model used in the code:
  - Input Layer: The input shape in the ResNet50 model is defined as (32, 32, 3), indicating that it expects images with a size of 32x32 pixels to have three color channels (RGB).
  - Convolutional Layers: ResNet50 includes a range of convolutional layers, involving convolutional blocks containing residual connections. They extract keys from the input images at various abstraction levels.
  - Pooling Layers: Utilizing max pooling layers, ResNet50 aims to downsample the feature maps produced by the convolutional layers. Max pooling helps minimize the spatial dimensions of the feature maps while still preserving significant information.
  - Fully Connected Layers: The fully connected layers, also recognized as the dense layers, are not utilized straightforwardly in the presented code. Rather than that, the output from the last convolutional layer of ResNet50 is flattened (Flatten()) to establish a one-dimensional array, which is subsequently handed over to the custom dense layers implanted on top for classification.
  - Base Model (ResNet50): The ResNet50 model itself containers many layers, including convolutional layers, pooling layers, and residual blocks. However, as it's instantiated with include_top = False, only the convolutional layers are added to the model. The

number of layers in the ResNet50 base model can be obtained using the summary() method.

- Flatten Layer: After the ResNet50 base model, a Flatten layer is added, which does not add any new trainable parameters. It simply flattens the output of the ResNet50 base model into a one-dimensional array.
- Dense Layers: Two dense layers are added on top of the Flatten layer. The first dense layer containers 512 units, and the second dense layer contains 43 units (the number of classes in the dataset).
- Dropout Layer: A Dropout layer is a rare bird, with a dropout rate of 0.5, added after the first dense layer.

- In short, this ResNet50 architecture does not directly manipulate or modify the internal layers of the ResNet50 model. Instead, it leverages the pre-trained ResNet50 model as a fixed feature extractor, extracting relevant features from the input image, which are then fed into custom dense layers for classification purposes.

CHAPTER - 8

PERFORMANCE EVALUATION & RESULT ANALYSIS

S.No.	Model	Evaluation Metrics
1.	CNN	Accuracy: 0.98 Weighted Precision: 0.97 Weighted F1-score: 0.98 Weighted Recall: 0.95 No. of Epochs: 30
2.	ResNet50	Test Accuracy: 0.93 Test Loss: 0.43 Precision: 0.91 Recall: 0.90 F1-score: 0.90 No. of Epochs: 100

Table 1: Performance metrics

8.1 CNN Model METRICS:

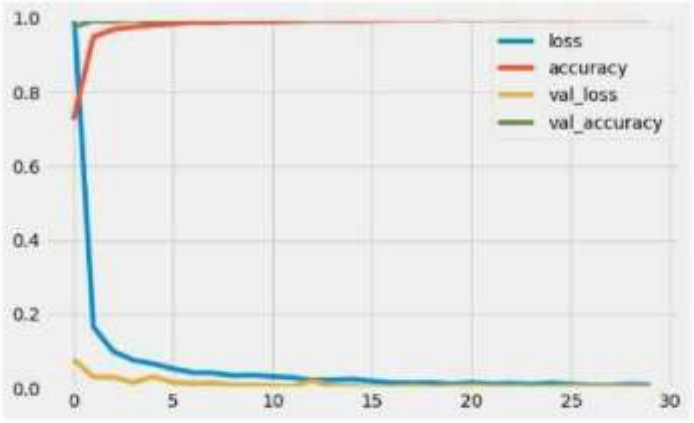


Fig. 7: CNN Model Metrics

19

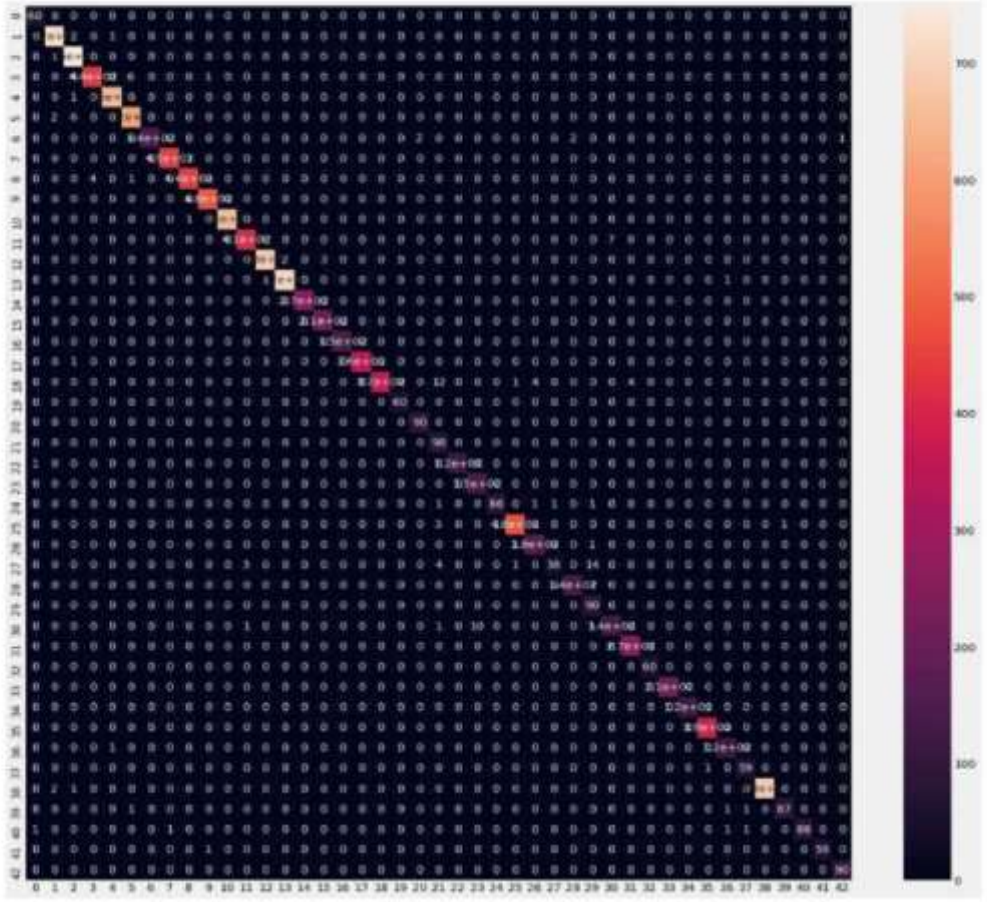


Fig. 8: CNN model Heatmap

Fig. 9: CNN model output

8.2 ResNet50 Model METRICS:

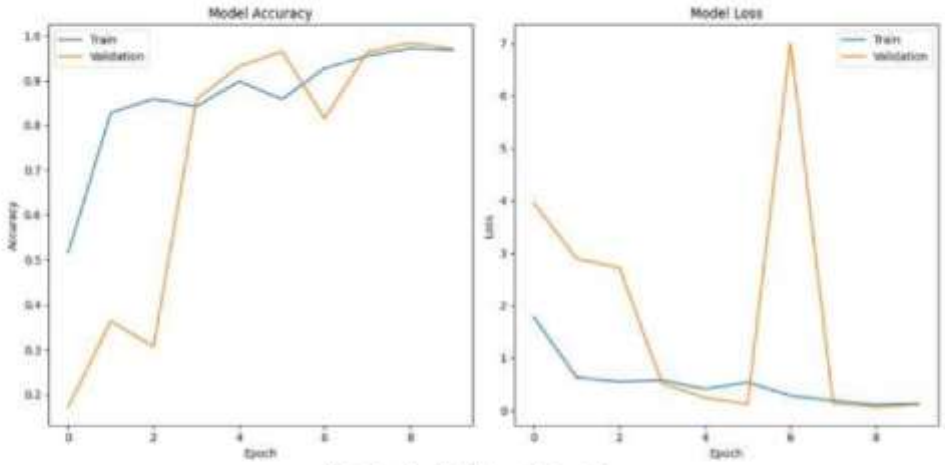


Fig. 10: ResNet50 model metrics

Fig. 11: ResNet50 model output

RESULTS COMPARISION

Approach Used	Accuracy	Precision	Recall	F1-score
CNN	0.98	0.97	0.95	0.98
ResNet50	0.93	0.91	0.90	0.90
YOLOv5	0.88	0.82	0.79	-
VSSA-net	0.94	0.78	-	-
Alexnet	0.92	0.90	-	-
Faster RCNN	0.92	0.93	0.94	0.93

Table 2: Results comparison

CHAPTER - 9

CONCLUSION & FUTURE SCOPE

In conclusion, this project addresses the critical challenges faced by autonomous vehicles in accurately detecting traffic signs, especially in adverse weather conditions. By building upon insights from prior research and acknowledging limitations associated with small datasets and varied weather scenarios, our proposed solution employs a comprehensive approach. Through the integration of benchmark datasets, innovative data augmentation techniques, and the utilization of state-of-the-art deep learning models such Convolution Neural Network (CNN), and ResNet50, we aim to significantly enhance the robustness and accuracy of traffic sign detection. This endeavor not only contributes to overcoming existing limitations but also paves the way for more reliable and safer autonomous navigation, ultimately advancing the field of autonomous driving technology.

The future scope of this project extends to various avenues, including the integration of real-time data streams for dynamic adaptation to changing weather conditions, refinement of the deep learning models for improved accuracy and efficiency, and exploration of multi-sensor fusion techniques to further enhancement the robustness of the traffic sign detection system. Additionally, the insights gained from this project can be applied to other domains requiring resilient perception systems, such as surveillance, robotics, and augmented reality, thereby contributing to the broader advancement of computer vision technology.

ORIGINALITY REPORT

16%
SIMILARITY INDEX

PRIMARY SOURCES

1	github.com Internet	174 words — 5%
2	blog.paperspace.com Internet	53 words — 1%
3	assets.researchsquare.com Internet	47 words — 1%
4	Vikas Khare, Ankita Jain. "Predict the Performance of Driverless Car through the Cognitive Data Analysis and Reliability Analysis based Approach", e-Prime - Advances in Electrical Engineering, Electronics and Energy, 2023 Crossref	25 words — 1%
5	paperswithcode.com Internet	25 words — 1%
6	Dalia Ezzat, Aboul Ella Hassanien, Mohamed Hamed N. Taha, Siddhartha Bhattacharyya, Snasel Vaclav. "Chapter 74 Transfer Learning with a Fine-Tuned CNN Model for Classifying Augmented Natural Images", Springer Science and Business Media LLC, 2020 Crossref	23 words — 1%
7	Islam, Kh Tohidul. "Real-Time Vision-Based Malaysian Road Sign Recognition Using an Artificial Neural Network", University of Malaya (Malaysia), 2023 ProQuest	20 words — 1%

8	Y. Jeevan Nagendra Kumar, Sukanya Ledalla, Avvari Pavithra, G. Vijendar Reddy, Rachana Joshi, L. Jayahari. "DFR-TSD: A Sustainable Deep Learning Based Framework for Sustainable Robust Traffic Sign Detection under Challenging Weather Conditions", E3S Web of Conferences, 2023 Crossref	19 words — 1%
9	www.mdpi.com Internet	19 words — 1%
10	www.gamesradar.com Internet	18 words — < 1%
11	Bhatt, Rashmi. "A Cross Sectional Study to Develop a Scale on Health Related QOL for Assessing the Effect of Unhealthy Food Habits in Apparently Healthy Subjects", Rajiv Gandhi University of Health Sciences (India), 2023 ProQuest	16 words — < 1%
12	www.ijnrd.org Internet	16 words — < 1%
13	qubixity.net Internet	15 words — < 1%
14	ijarcce.com Internet	14 words — < 1%
15	Abdulrahman Alkojak Almansi, Sima Sugarova, Abdulrahman Alsanosi, Fida Almuhawas et al. "A novel radiological software prototype for automatically detecting the inner ear and classifying normal from malformed anatomy", Computers in Biology and Medicine, 2024 Crossref	11 words — < 1%

16	www.ijera.com Internet	11 words — < 1%
17	speakerdeck.com Internet	10 words — < 1%
18	www.nature.com Internet	10 words — < 1%
19	www.scilit.net Internet	10 words — < 1%
20	"Soft Computing and Signal Processing", Springer Science and Business Media LLC, 2019 Crossref	9 words — < 1%
21	Shakil Ahmed Sumon, Raihan Goni, Niyaz Bin Hashem, Tanzil Shahria, Rashedur M. Rahman. "Violence Detection by Pretrained Modules with Different Deep Learning Approaches", Vietnam Journal of Computer Science, 2019 Crossref	8 words — < 1%
22	Xinhua Wu, Hongjun Jia. "Building Crack Identification and Total Quality Management Method Based on Deep Learning", Pattern Recognition Letters, 2021 Crossref	8 words — < 1%
23	medium.com Internet	8 words — < 1%
24	scholarworks.uaeu.ac.ae Internet	8 words — < 1%
25	Ozen, Elbruz. "Algorithm-Centric Design of Reliable and Efficient Deep Learning Processing Systems",	7 words — < 1%

University of California, San Diego, 2023
ProQuest

26 digital.library.ncat.edu
Internet

7 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	OFF	EXCLUDE MATCHES	OFF