



# **Smart Road Guard: Enabling Intelligent Road Sign Recognition and Real-Time Safety Guidance System**

**Name: Mayuri Bhosale**

**Student ID: 10640089**

Applied Research Project submitted in partial fulfilment of the requirements for the degree of  
Masters of Science in Data Analytics  
at Dublin Business School

**Supervisor: Mr. Ankush Arya**

**(January, 2024)**

## **Declaration**

I declare that this Applied Research Project that I have submitted to Dublin Business School for the award of Masters of Science in Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.

Signed: Mayuri Bhosale

Student Number: 10640089

Date: 08/01/2024

## **Acknowledgement**

I extend my heartfelt gratitude to everyone who supported and contributed to the completion of this dissertation.

I am immensely thankful to Mr. Ankush Arya, whose invaluable guidance, expertise, and unwavering support have been instrumental throughout this research journey. Their mentorship and insightful feedback shaped the direction and quality of this study, paving the way for its successful completion.

Furthermore, my gratitude extends to the Dublin Business School for providing access to resources, facilities, and the conducive academic environment that fostered intellectual growth and learning.

Lastly, my heartfelt thanks to my family and friends for their unwavering encouragement, understanding, and continuous motivation throughout this academic pursuit.

Their collective support and encouragement have been indispensable in making this research endeavor a reality.

## **Abstract**

Road safety is a critical aspect of modern transportation systems, necessitating intelligent solutions to enhance recognition and guidance mechanisms. This thesis presents an in-depth exploration of Convolutional Neural Networks (CNNs) as a foundational technology for developing an advanced road sign recognition system with real-time safety guidance capabilities. The research methodology begins with the collection of a representative dataset obtained from Google, forming the basis for subsequent analyses. Various image preprocessing and augmentation techniques are systematically applied to enrich the dataset, ensuring robust model training. The PyTorch framework is employed for model development, utilizing a base CNN architecture to establish a foundational understanding of road sign recognition. In pursuit of improved performance, transfer learning techniques, specifically leveraging MobileNetV2, are investigated. This involves adapting a pre-trained model to the road sign recognition domain, enhancing the model's ability to generalize across diverse scenarios. A comprehensive analysis of the developed models follows, incorporating metrics such as accuracy, precision, recall, and F1 score. The evaluation considers the effectiveness of each model in recognizing road signs and providing real-time safety guidance. Results reveal insights into the strengths and limitations of the models under various conditions. The study concludes with predictions on test data points, validating the real-world applicability of the developed models. The findings contribute to the broader discourse on intelligent road safety systems, emphasizing the significance of CNNs and transfer learning in advancing road sign recognition and safety guidance technologies.

This thesis aims to not only present technological solution but also contribute to theoretical understanding of intelligent transportation systems, future research and practical applications in realm of road safety and traffic management. Implementation of this research aligns with ever-growing demand for efficient and adaptive solutions in field of intelligent transportation. Utilization of transfer learning, particularly MobileNetV2, showcases significant enhancement in model's ability to discern and interpret complex road sign patterns. Fusion of advanced deep learning techniques, comprehensive dataset handling, and real-time guidance capabilities positions the proposed system as viable and promising solution for improving road safety and traffic management in our ever-evolving urban landscapes. The journey from data collection to model evaluation provides roadmap for future advancements, emphasizing continuous evolution of intelligent systems for the benefit of society.

**Keywords:**

Road sign recognition, Convolutional Neural Networks (CNNs), Real-time safety guidance, Intelligent transportation systems, Transfer learning, Deep learning, Traffic management, Image preprocessing, MobileNetV2, Road safety technologies.

## INDEX

### Table of Contents:

Declaration .....	1
Chapter 1: Introduction .....	7
1.1 Introduction:.....	7
1.2 Background:.....	8
1.3 Motivation: .....	11
1.4 Research Aim:.....	12
1.5 Research Objectives: .....	12
1.6 Research Question: .....	13
1.7 Research Outline: .....	14
Chapter 2: Literature Review .....	16
2.1 Summary and Gaps in the Existing Literature:.....	24
2.2 Related Work Done: .....	26
2.3 My Contribution: .....	27
Chapter 3: Design and Methodology .....	28
3.1 Design:.....	28
3.2 Methodology:.....	29
Chapter 4 - Implementation .....	31
Chapter 5: Discussion.....	49

5.1 Discussion:.....	49
5.2 Challenges: .....	51
5.3 Future Scope: .....	53
Chapter 6: Conclusion.....	55
Chapter 7: References.....	56
Chapter 8: APPENDIX .....	60

## Chapter 1: Introduction

### 1.1 Introduction:

The landscape of modern transportation is during a profound transformation, propelled by the intersection of urbanization, technological advancements, and the ever-increasing demands placed on our road infrastructure. The dynamic interplay of these factors is reshaping the way we perceive, navigate, and manage our roadways. Urbanization, marked by the relentless growth of cities and the swelling tide of urban populations, is a key driver in this paradigm shift. The sprawling expansion of urban areas brings forth intricate challenges for transportation systems, demanding innovative solutions to accommodate the burgeoning needs of commuters and the efficient flow of traffic. Simultaneously, technological advancements are ushering in a new era of possibilities for transportation. From smart traffic management systems to the integration of artificial intelligence, technology is becoming an integral part of the transportation ecosystem. These innovations hold the promise of optimizing traffic flow, reducing congestion, and ultimately enhancing the overall efficiency of transportation networks.

However, this surge in urbanization and technological integration amplifies the demands on our road infrastructure. The increasing complexity of road networks, coupled with a growing diversity of transportation modes, necessitates a reevaluation of traditional approaches to road safety and traffic management. Amidst these transformative forces, the imperative to enhance road safety and traffic management emerges as a central concern. It is within this evolving landscape that road signs assume a pivotal role. These silent communicators line our roads, providing crucial information, guidance, and regulations to drivers. In essence, road signs are the unsung heroes of traffic management, quietly but effectively shaping the



behaviour of drivers and ensuring the smooth orchestration of vehicles through the intricate dance of intersections and highways. The silent yet potent influence of road signs is magnified in the context of enhancing road safety. As the conduits of critical information, these signposts contribute to the prevention of accidents, the reduction of traffic violations, and the overall improvement of road safety. In the face of a changing transportation paradigm, understanding and optimizing the role of road signs becomes imperative for ensuring not only the efficient movement of traffic but also the safety of all road users.

Therefore, this thesis embarks on a journey to explore how advanced technologies, particularly Convolutional Neural Networks (CNNs), can be harnessed to augment the efficacy of road sign recognition. By delving into the intricacies of these silent communicators, we aim to contribute to the ongoing efforts to enhance road safety and traffic management in the ever-evolving landscape of modern transportation systems.

## **1.2 Background:**

Road signs constitute a fundamental aspect of our transportation infrastructure, serving as silent communicators that convey essential information to drivers navigating the intricate web of roadways. These signs, characterized by a diverse array of shapes, colours, and symbols, play a pivotal role in regulating and guiding traffic flow. They act as unobtrusive instructors, providing vital information that ranges from speed limits and directional guidance to warnings about potential hazards. The accurate and timely recognition of these road signs is paramount for ensuring safe navigation and compliance with traffic regulations. As drivers traverse diverse urban and rural landscapes, road signs serve as beacons of guidance, aiding in decision-making and contributing to the overall safety of the road ecosystem.

However, the contemporary transportation milieu presents a growing set of challenges to conventional methods of road sign recognition. The complexity of modern road networks, characterized by a multitude of intersections, varying lane configurations, and diverse traffic scenarios, demands sophisticated recognition systems capable of interpreting a rich tapestry of visual information. Moreover, environmental conditions further complicate this task, as factors such as adverse weather, low lighting, and obstructions can hinder the visibility of road signs. Conventional methods, reliant on rule-based algorithms or simplistic image processing, may struggle to cope with the intricacies of real-world scenarios. The need for adaptive, context-aware systems becomes evident in the face of these challenges. The advent of advanced technologies, particularly the application of machine learning and deep learning methodologies, offers a promising avenue for overcoming these hurdles.

In this context, the research endeavours to explore the potential of Convolutional Neural Networks (CNNs) as a sophisticated tool for road sign recognition. By harnessing the power of deep learning, we aim to develop a system that can discern and interpret the diverse visual cues presented by road signs. This exploration not only addresses the immediate challenge of accurate recognition but also lays the foundation for the integration of real-time safety guidance, thereby contributing to the overarching goals of enhanced road safety and efficient traffic management in the face of a dynamically evolving transportation landscape. The evolving challenges within transportation systems, characterized by the burgeoning complexities of road networks and the escalating demands on road safety, have spurred a crucial need for the integration of advanced technologies. In this era of technological acceleration, the imperative to enhance the capabilities of transportation systems has

become paramount. Among the pantheon of technological innovations, Convolutional Neural Networks (CNNs) have risen as a transformative force, particularly in the domain of image recognition. The intricacies of road sign recognition demand a level of sophistication that traditional methods struggle to achieve. Conventional rule-based algorithms and simplistic image processing techniques, while effective in controlled environments, often falter when confronted with the multifaceted challenges presented by real-world road scenarios. In response to this gap in capability, the integration of CNNs into transportation systems has become not just desirable but imperative. CNNs are a class of deep learning models specifically designed to process and interpret visual data. Their architecture is inspired by the visual processing hierarchy of the human brain, allowing them to automatically learn hierarchical features from images. This intrinsic ability to discern and understand patterns at various levels of complexity positions CNNs as an ideal tool for addressing the intricate and context-dependent nature of road sign recognition. The transformative power of CNNs lies in their capacity to autonomously learn abstract features from vast datasets, obviating the need for explicit programming or rule-setting. This attribute is particularly beneficial in the realm of road sign recognition, where signs exhibit diverse shapes, colours, and symbols, and contextual interpretation is crucial for accurate understanding.

This thesis seeks to delve into the nuances of intelligent road sign recognition, exploring how CNNs can be effectively utilized not only to identify road signs accurately but also to integrate this information into real-time safety guidance systems. By delving into the depths of data collection, preprocessing, model development, and evaluation, this research aims to contribute to the ongoing discourse on the application of deep learning techniques to enhance road safety and traffic management.

### **1.3 Motivation:**

The motivation propelling this thesis is rooted in the pressing need to address escalating concerns regarding road safety and traffic management in the contemporary world. The surge in the global number of vehicles has led to alarming rates of traffic accidents, injuries, and fatalities, causing not only immense human suffering but also imposing substantial economic burdens, overwhelming healthcare systems, and disruptive consequences for society at large. Traditional methods of traffic management, primarily reliant on static road signs and signals, prove inadequate in handling the complexities and dynamic nature of modern traffic conditions. These conventional systems are static, incapable of adapting to real-time situations, often falling short in effectively preventing accidents and guiding drivers through evolving traffic scenarios. To confront these challenges, there is a critical need to explore innovative solutions harnessing cutting-edge technologies, such as artificial intelligence, computer vision, and sensor systems. An intelligent road sign recognition and real-time safety guidance system emerges as a transformative approach to road safety by providing drivers with real-time information. This approach aims to reduce the incidence of accidents, enhance traffic flow, and ultimately save lives.

This research project is motivated by a profound commitment to addressing these critical issues and instigating positive change in the realm of road safety and traffic management. Through the exploration, design, and implementation of an intelligent system with the potential to revolutionize the way we navigate our roadways, our aim is to make a meaningful contribution to mitigating the global road safety crisis. By doing so, we aspire to save lives, diminish economic losses, and improve the overall quality of life for individuals and communities impacted by traffic accidents. Through comprehensive research, innovative

technological solutions, and the dedication of researchers and professionals in the field, this thesis seeks to bring us closer to a future where road safety is significantly improved, accidents are minimized, and traffic management is more efficient. It is our firm belief that this work can contribute to building a safer and more secure transportation environment for all.

### **1.3 Research Aim:**

Develop and evaluate an intelligent road sign recognition system integrated with real-time safety guidance to enhance road safety and traffic management in dynamic environments.

### **1.4 Research Objectives:**

- **Develop an Advanced Deep Learning Model:** Create a sophisticated deep learning model capable of accurately classifying various road conditions such as curves, straight roads, junctions, and uneven roads.
- **Training with Diverse Road Image Dataset:** Train the model using a diverse dataset comprising different road conditions to enable the system to recognize and classify distinguishing features associated with each condition.
- **Real-time Recognition and Predictions:** Integrate the trained model into a system capable of real-time analysis using image inputs from vehicle-mounted sensors or cameras.
- **Road Sign and Caution Symbol Deployment:** Enable the system to provide immediate recognition of road signs and caution symbols based on the predicted road conditions.

- **Enhanced Driver Awareness and Road Safety:** Evaluate the system's effectiveness in enhancing driver awareness by providing timely guidance and road sign suggestions, contributing to improved road safety.

### **1.6 Research Question:**

"How can Convolutional Neural Networks (CNNs) be effectively employed to develop an intelligent road sign recognition system that not only accurately identifies road signs but also offers real-time safety guidance to drivers, aiming to enhance road safety and optimize traffic management?"

This research question encapsulates the core focus of the thesis, which is the exploration of CNNs as a solution to the problem of road sign recognition and real-time safety guidance. It addresses the need to create a system that not only identifies road signs but also leverages this information to actively guide drivers, thereby improving road safety and traffic management. The study will investigate the design, implementation, and evaluation of such a system to assess its efficacy in achieving these objectives.

## **1.7 Research Outline:**

### **Chapter 1 - Introduction:**

Introduces urgent need for advanced technologies in addressing rising road safety concerns, outline structure of thesis to explore intelligent road sign recognition for enhanced traffic management.

### **Chapter 2 - Background:**

Examines the critical role of road signs in traffic regulation, highlights challenges in current systems, and explores evolution of transportation systems and the integration of advanced technologies.

### **Chapter 3 - Design and Methodology:**

Details research design, dataset collection method, application of image preprocessing, transfer learning techniques for model development, emphasizing use of PyTorch, CNN architectures.

### **Chapter 4 - Implementation and Evaluation:**

Showcases implementation process, training details, and performance evaluation metrics, comparing efficacy of base CNN and transfer learning models, with focus on real-world applicability through predictions on test data.

#### Chapter 5 - Discussion:

Interprets results, compares findings with previous studies, discusses limitations and implications, explores future research directions, and evaluates the practical applications and integration challenges of the developed system.

#### Chapter 6 - Conclusion:

Summarizes key findings, emphasizes contributions to the field, and offers concluding remarks, underlining the potential impact of the research on road safety and traffic management.

#### Chapter 7 - References:

Lists citations and references in the appropriate style, providing a comprehensive bibliography for the research.



## Chapter 2: Literature Review

### *Introduction to Road Sign Recognition:*

The study of road sign recognition has garnered significant attention in the academic and research domains, reflecting its crucial role in the broader context of traffic management and road safety (Fregin et al., 2017; Biswas et al., 2014). The evolution of this field is underscored by the need to develop effective systems capable of autonomously identifying and interpreting road signs, contributing to safer and more efficient traffic flow. The early stages of research in road sign recognition predominantly leaned on rule-based algorithms and traditional image processing techniques as foundational methodologies (Chen & Lu, 2016). Rule-based algorithms formed the backbone of initial approaches to road sign recognition. These algorithms operated on predefined sets of rules and patterns, attempting to match observed features in the input data with predetermined criteria (Creusen et al., 2012). While suitable for certain controlled environments, rule-based systems encountered challenges in adapting to the dynamic and diverse nature of real-world traffic scenarios. Moreover, these approaches were often limited by their inability to handle the complexities presented by a wide array of road sign shapes, colours, and environmental conditions (Handoko et al., 2021). Traditional image processing techniques also played a fundamental role in the early development of road sign recognition systems. These techniques involved the extraction of specific features from images, often relying on colour segmentation, edge detection, and template matching (Fifik & Turán, 2010). However, the effectiveness of these methods was constrained by their reliance on manually crafted features and predetermined thresholds, making them less adaptable to the variability inherent in real-world traffic environments (Houben et al., 2013). As traffic management systems evolved, the limitations of rule-based

algorithms and traditional image processing techniques became apparent (Islam & Raj, 2017). The growing complexity of road networks, the diversity of road sign designs, and the need for systems capable of real-time adaptation prompted a shift towards more sophisticated and adaptive approaches. This shift laid the groundwork for the exploration of advanced technologies, such as artificial intelligence and computer vision, with a particular emphasis on the transformative role of Convolutional Neural Networks (CNNs) in redefining the landscape of road sign recognition (Zhang et al., 2020; Lee & Kim, 2018).

#### *Challenges in Conventional Traffic Management Systems:*

The examination of conventional traffic management systems reveals inherent limitations that have become a focal point in scholarly studies. These systems, characterized by their static nature, face significant challenges in adapting to the dynamic and real-time conditions prevalent in modern traffic scenarios (Houben et al., 2013). Studies consistently emphasize the static nature of traditional traffic management systems as a critical drawback. These systems rely on predetermined rules and structures, designed to regulate traffic based on predefined parameters. However, this rigidity poses a considerable obstacle, particularly in the face of the dynamic nature of today's traffic, characterized by ever-changing variables such as varying traffic volumes, unexpected incidents, and diverse road user behaviours (Author11, Year11). One of the notable shortcomings of static traffic management is its inability to dynamically respond to real-time conditions. The lack of adaptability hinders the system's capacity to efficiently allocate resources, optimize traffic flow, and promptly address emerging issues. This inadequacy is particularly evident in scenarios where road conditions change rapidly, such as during incidents, adverse weather, or unexpected surges in traffic volume (Lee & Kim, 2018).

The recognition of these challenges has fuelled a growing interest in the exploration of advanced technologies to overcome the limitations of conventional traffic management systems. The inefficiencies of static systems have prompted researchers and practitioners to seek innovative solutions that leverage the capabilities of artificial intelligence, machine learning, and real-time data processing (Author13, Year13). The objective is to create systems that can dynamically adapt to evolving traffic conditions, enhancing overall efficiency, safety, and responsiveness in managing complex traffic scenarios. In summary, the literature consistently underscores the challenges posed by the static nature of conventional traffic management systems, motivating a shift towards advanced technologies capable of addressing the dynamic complexities inherent in modern traffic scenarios. This shift reflects a broader paradigm in traffic management philosophy, aiming to transform systems from static rule-based approaches to adaptive, real-time solutions that can effectively navigate the intricacies of contemporary traffic environments.

#### *Role of Advanced Technologies in Traffic Management Systems:*

The literature has been a rich source of exploration into the integration of advanced technologies, particularly artificial intelligence (AI) and computer vision, within traffic management systems. Researchers and practitioners alike recognize the transformative potential of these technologies in addressing the evolving challenges of modern traffic scenarios, as highlighted in various studies (Fregin et al., 2017; Changzhen et al., 2017; Biswas et al., 2014). Artificial intelligence, encompassing machine learning and deep learning techniques, has emerged as a powerful tool in enhancing the capabilities of traffic management systems. The ability of AI to process vast amounts of data, learn intricate patterns, and adapt to changing conditions aligns seamlessly with the dynamic nature of

contemporary traffic environments (Islam & Raj, 2017). This adaptability is particularly crucial in optimizing traffic flow, managing congestion, and improving overall system responsiveness. Within the realm of computer vision, Convolutional Neural Networks (CNNs) have garnered substantial attention for their prowess in image recognition tasks, including road sign detection. The application of CNNs in traffic management systems has been instrumental due to their ability to automatically learn hierarchical features from visual data (Changzhen et al., 2017). CNNs excel in discerning intricate patterns and representations within images, making them well-suited for tasks such as road sign recognition, an integral component of efficient traffic management.

The work by Fregin et al. (2017) emphasizes the effectiveness of stereo vision, a component of computer vision, for traffic light recognition. This highlights the broader spectrum of applications within traffic management where advanced vision technologies contribute to enhanced situational awareness and decision-making. The study conducted by Biswas et al. (2014) delves into the detection and classification of speed limit traffic signs. The incorporation of computer vision techniques, as presented in this study, showcases the versatility of these technologies in addressing specific challenges related to road sign recognition and regulation within traffic systems. In essence, the literature underscores the pivotal role of advanced technologies, particularly AI and computer vision, in revolutionizing traffic management systems. The adaptability, learning capabilities, and efficiency of these technologies, exemplified by the prominence of CNNs, signify a paradigm shift towards intelligent and data-driven approaches in optimizing traffic flow, enhancing road safety, and addressing the intricate challenges of modern traffic environments.

*Evolution of Road Sign Recognition Techniques:*

The evolution of road sign recognition techniques represents a dynamic trajectory in response to the increasing complexities of modern traffic scenarios. Early methods, often reliant on simplistic algorithms, have given way to more sophisticated techniques that leverage advanced technologies, as evidenced in studies such as those by Fregin et al. (2017) and Biswas et al. (2014). Fregin et al. (2017) introduced a notable advancement by incorporating stereo vision into road sign recognition, specifically for traffic light recognition. Stereo vision, a technique that involves processing images from two or more cameras to create a three-dimensional representation, enhances the depth perception of the system. This innovation addresses the challenges of accurately recognizing and interpreting traffic lights, a critical aspect of comprehensive road sign recognition systems. The study by Fregin et al. exemplifies a shift towards incorporating more sophisticated sensing mechanisms to improve the accuracy and reliability of road sign recognition in diverse traffic scenarios. Biswas et al. (2014) contributed to the evolution of road sign recognition by focusing on the detection and classification of speed limit traffic signs. This study recognized the need for specialized techniques to handle different types of road signs, acknowledging the diverse nature of signage on roads. By delving into the specifics of speed limit sign detection, the research by Biswas et al. demonstrated a targeted approach to improving the functionality and effectiveness of road sign recognition systems. This shift towards specialized detection methods aligns with the broader trend of tailoring recognition techniques to address the unique challenges posed by different categories of road signs.

These advancements highlight a progression from rudimentary road sign recognition methods to more nuanced and specialized techniques. The integration of stereo vision and the targeted focus on specific road sign categories showcase a growing recognition of the need for

comprehensive and adaptable recognition systems that can navigate the intricacies of modern traffic environments. As road sign recognition continues to evolve, the integration of such sophisticated techniques becomes integral to enhancing overall system performance and contributing to the broader goal of efficient and safe traffic management.

#### *Application of Deep Learning in Traffic Sign Recognition:*

The application of deep learning techniques, notably Convolutional Neural Networks (CNNs), has marked a significant breakthrough in the domain of traffic sign recognition. The efficacy of deep learning, particularly CNNs, is underscored by studies conducted by Changzhen et al. (2017) and Chen and Lu (2016), both of which highlight the remarkable success achieved in the accuracy and efficiency of detecting and classifying various traffic signs. Changzhen et al. (2017) demonstrated the effectiveness of CNNs in traffic sign recognition by emphasizing their ability to automatically learn hierarchical features from visual data. The study delves into the intricacies of a traffic sign detection algorithm based on a deep convolutional neural network. By leveraging the hierarchical feature learning capabilities of CNNs, the algorithm showcased a high degree of accuracy in detecting and classifying diverse traffic signs. This signifies a departure from traditional methods, indicating that CNNs can adapt and generalize well to the complexities presented by a wide array of road sign shapes, colors, and environmental conditions.

Similarly, Chen and Lu (2016) contributed to the exploration of deep learning in traffic sign recognition by introducing a method that combines discriminative Adaboost and support vector regression. The study emphasizes the accuracy and efficiency achieved through the integration of CNNs, particularly in the context of detecting and classifying traffic signs. The utilization of CNNs in this hybrid approach showcases their versatility in improving the

robustness and precision of traffic sign recognition systems. The success of CNNs in these studies can be attributed to their inherent ability to automatically learn and extract intricate features from visual data. CNNs excel in discerning complex patterns within images, a quality that is crucial for accurate and efficient recognition of traffic signs in various real-world scenarios. The hierarchical feature learning architecture of CNNs allows them to capture both low-level and high-level features, making them well-suited for tasks that require a nuanced understanding of visual information.

In summary, the application of deep learning, particularly through the utilization of CNNs, has proven to be a game-changer in the realm of traffic sign recognition. The studies by Changzhen et al. and Chen and Lu underscore the accuracy, efficiency, and adaptability of CNNs in detecting and classifying a diverse range of traffic signs, marking a paradigm shift towards more effective and robust recognition systems in the field of traffic management.

#### *Enhancements through Transfer Learning:*

The literature suggests that transfer learning, as demonstrated by Handoko et al. [7] and Lee and Kim [11], has become a valuable approach in adapting pre-trained models like MobileNetV2 to the specific domain of road sign recognition.

#### *Real-Time Safety Guidance Systems:*

The integration of real-time safety guidance systems alongside road sign recognition represents a pivotal advancement in enhancing the effectiveness of traffic management. Studies conducted by Islam and Raj (2017) and Yuan Yuan et al. (2019) have contributed significantly to this area by exploring the integration of systems that actively guide drivers based on identified road signs. Islam and Raj (2017) investigated the development of a real-

time vision-based road sign recognition system that goes beyond mere identification. The study emphasizes the importance of not only recognizing road signs, but also actively guiding drivers based on the identified signs. This integrated approach aims to provide immediate and contextually relevant guidance to drivers, contributing to enhanced road safety. By incorporating an artificial neural network for real-time road sign recognition, the system demonstrated the potential to dynamically influence driver behaviour through timely guidance.

Similarly, Yuan Yuan et al. (2019) delved into the realm of real-time safety guidance by proposing the VSSA-net (Vertical Spatial Sequence Attention Network) for traffic sign detection. The study extends beyond the traditional scope of road sign recognition by integrating a system that not only identifies signs but also provides guidance in real-time. By incorporating spatial attention mechanisms, the VSSA-net demonstrated improved accuracy in detecting and classifying traffic signs, laying the groundwork for effective real-time safety guidance. The integration of real-time safety guidance systems represents a paradigm shift in traffic management philosophy. Traditionally, road sign recognition focused on passive identification without immediate impact on driver behaviour. The studies by Islam and Raj, as well as Yuan Yuan et al., recognize the need for a more proactive approach to road safety. These systems leverage the identified road signs as actionable information, actively guiding drivers to promote safer driving practices. The significance of real-time safety guidance systems lies in their potential to prevent accidents, reduce traffic violations, and enhance overall road safety. By dynamically influencing driver behaviour based on the recognized road signs, these systems contribute to a more responsive and adaptive traffic management environment. As traffic conditions evolve, real-time safety guidance systems provide a crucial



layer of support, aligning road sign recognition with practical interventions that actively contribute to a safer and more efficient road network.

#### *Evaluation Metrics and Benchmarking:*

Efforts have been made to establish benchmark datasets and evaluation metrics. Houben et al.'s work [8] on the German Traffic Sign Detection Benchmark and the comprehensive analysis by Zhang et al. [10] provide insights into performance evaluation criteria for traffic sign recognition models.

### **2.1 Summary and Gaps in the Existing Literature:**

In synthesizing the existing literature, several key themes emerge, providing insights into the evolution of road sign recognition, the significance of advanced technologies, and the burgeoning importance of real-time safety guidance systems. However, a critical gap persists in comprehending the real-world applicability and the holistic integration of these systems within dynamic traffic environments. The evolution of road sign recognition techniques, as evidenced by studies such as those conducted by Fregin et al. (2017) and Biswas et al. (2014), highlights a trajectory from early methods to more sophisticated approaches. The integration of stereo vision and the targeted focus on specific road sign categories exemplify a concerted effort to enhance accuracy and adaptability of recognition systems. The literature underscores the need for systems that can navigate the intricate challenges posed by the diversity of road signs and dynamic traffic conditions. The role of advanced technologies, particularly artificial intelligence, and computer vision, stands out as a transformative force in optimization of traffic management systems. The application of deep learning, notably

through Convolutional Neural Networks (CNNs), has shown remarkable success in traffic sign recognition (Changzhen et al., 2017; Chen & Lu, 2016). These technologies contribute to the adaptability, learning capabilities, and efficiency of traffic management systems, aligning with the dynamic nature of modern traffic scenarios. Real-time safety guidance systems represent paradigm shift in the traditional scope of road sign recognition. Studies by Islam and Raj (2017) and Yuan Yuan et al. (2019) introduce concept of actively guiding drivers based on identified road signs. This marks departure from passive identification to more proactive approach, aiming to influence driver behaviour in real-time. The integration of these systems offers potential to significantly enhance road safety and contribute to more responsive traffic management environment.

However, amidst these advancements, notable gap exists in understanding practical applicability and holistic integration of these systems in real-world, dynamic traffic environments. While literature provides insights into technical aspects of recognition and guidance systems, there is need for research that bridges gap between theoretical advancements and their effective implementation in diverse and unpredictable traffic scenarios. The real-world applicability of these technologies, their robustness in handling complex scenarios, and their seamless integration into existing traffic infrastructure represent areas where further research is warranted. Additionally, literature suggests need for studies that explore human factors involved, such as driver response and interaction with these advanced systems, to ensure their effectiveness and acceptance in practical traffic management scenarios. Closing this gap is crucial for successful deployment and impact of these technologies in real-world traffic environments.

## 2.2 Related Work Done:

Previous researchers have made significant contributions to the field of road sign recognition and intelligent traffic management systems. Fregin et al. (2017) introduced stereo vision as a means of enhancing traffic light recognition, laying the groundwork for more accurate and robust recognition systems. This early work reflects a focus on improving the capabilities of recognition systems through advancements in sensing technologies (Fregin et al., 2017).

Similarly, Biswas et al. (2014) delved into the detection and classification of speed limit traffic signs, showcasing the need for specialized techniques in handling different categories of road signs. This work emphasized the importance of tailoring recognition systems to address the diverse range of road signs encountered in real-world traffic scenarios (Biswas et al., 2014).

Changzhen et al. (2017) and Chen and Lu (2016) significantly contributed to the application of deep learning techniques, particularly Convolutional Neural Networks (CNNs), in traffic sign recognition. These studies demonstrated the remarkable success achieved in accuracy and efficiency through the integration of CNNs, marking a paradigm shift in the approach to road sign recognition (Changzhen et al., 2017; Chen & Lu, 2016).

Islam and Raj (2017) and Yuan Yuan et al. (2019) expanded the scope of road sign recognition by integrating real-time safety guidance systems. These studies recognized the need for actively guiding drivers based on identified road signs, adding a crucial layer of proactiveness to traditional recognition systems (Islam & Raj, 2017; Yuan Yuan et al., 2019).

### **2.3 My Contribution:**

In the context of this research, my contribution builds upon the foundation laid by previous researchers. While past work has made significant strides in improving the accuracy and capabilities of road sign recognition systems, there remains a notable gap in understanding the real-world applicability and holistic integration of these systems within dynamic traffic environments. Our research aims to bridge this gap by focusing on the practical implementation and effectiveness of these systems in diverse and unpredictable traffic scenarios. I extend the existing literature by conducting comprehensive evaluations of road sign recognition systems in real-world conditions, considering factors such as varying environmental conditions, complex traffic scenarios, and the human element in driver behaviour. This approach ensures a more nuanced understanding of the challenges and opportunities presented by deploying intelligent traffic management systems. Our research places a particular emphasis on the seamless integration of real-time safety guidance systems with road sign recognition. By actively guiding drivers based on identified road signs, we aim to enhance the proactive role of these systems in influencing driver behaviour and, consequently, improving overall road safety.

In summary, my contribution lies in advancing the practical implementation of road sign recognition and intelligent traffic management systems, addressing the gaps identified in the existing literature. Through a holistic approach that considers real-world dynamics and the integration of real-time safety guidance, our research seeks to propel the field towards more effective and applicable solutions for modern traffic management challenges.

## Chapter 3: Design and Methodology

### 3.1 Design:

The chosen research design employs an experimental approach that utilizes Convolutional Neural Networks (CNNs) for classifying road images. This quantitative methodology focuses on evaluating the efficacy of CNNs in identifying various road conditions through image classification. The rationale behind selecting this design is its suitability for investigating image classification tasks, particularly in diverse road condition identification.

#### Data Collection:

The dataset utilized in this study comprises diverse road images acquired from multiple sources, including web scraping, public repositories, and crowd-sourced data. The aim of this method was to ensure dataset diversity and comprehensiveness, aligning with the study's objective of providing diverse samples for training and validation.

#### Experimental Setup:

The study's experimental setup involves the utilization of GPU-enabled systems and frameworks like PyTorch and TensorFlow for model training and validation. This setup was chosen due to its efficiency in handling deep learning algorithms and processing substantial image data. By leveraging these frameworks and hardware, the study aims to optimize computational resources and facilitate efficient model training and testing.

#### Variables and Parameters:

The critical variables in this study encompass various road conditions: left curve, pot holes, right curve, S-curved, snow-covered, and straight roads. These variables were measured through image analysis and subsequently manipulated for model training and testing. The study's focus on these variables aligns with the research objectives of identifying and classifying diverse road conditions through CNN-based models.

### **3.2 Methodology:**

#### Convolutional Neural Networks (CNNs):

The study employed CNNs due to their effectiveness in image classification tasks. This deep learning architecture, inspired by biological processes, is renowned for its capability to automatically learn intricate features from images. The choice of CNNs aligns with the study's objective of accurately identifying and classifying various road conditions depicted in images.

#### Data Preprocessing:

Data preprocessing involved several steps, including resizing images to a standard dimension, augmenting the dataset for increased diversity, and normalizing pixel values. This process aimed to enhance model performance by ensuring consistency in image size, introducing variability through augmentation, and standardizing pixel values for efficient training.

#### Model Training and Validation:

The dataset was divided into training and validation sets, maintaining a balance to prevent overfitting. The training set was utilized to optimize model parameters through iterations,

while the validation set facilitated the assessment of model generalization on unseen data. This iterative process involved adjusting hyperparameters, evaluating performance metrics, and refining the model architecture.

#### Evaluation Metrics:

The study measured the model's performance using various evaluation metrics, primarily focusing on accuracy, precision, recall, and F1 score. These metrics were employed to quantify the model's capability to correctly classify road conditions, considering both correct and incorrect predictions.

#### Transfer Learning:

Transfer learning using a pre-trained model (MobileNetV2) from ImageNet was explored. This technique leverages a pre-existing model's learned features, fine-tuning it for the road condition classification task. The rationale behind this approach was to capitalize on pre-trained knowledge and optimize model performance, especially with limited data.

These elements collectively constitute the methodology employed to conduct the study, emphasizing the utilization of CNNs, data preprocessing techniques, model training, evaluation, and the exploration of transfer learning strategies.

## Chapter 4 - Implementation

### 1. Data Collection and Preprocessing:

The process began with the collection of a diverse dataset containing various road conditions, including left curves, pot holes, right curves, S-curved roads, snow-covered paths, and straight roads. Following this, preprocessing steps were implemented to standardize images, such as resizing them to a uniform dimension, applying augmentation techniques, and normalizing pixel values. These steps ensured data uniformity and introduced variability to enhance model robustness.

The augmentation process involves generating multiple versions of each original image by applying a combination of transformations. The modifications are implemented following a sequential order defined within the code snippet. The resultant images carry filenames indicating the specific augmentation applied, allowing for easy identification.

In summary, the image augmentation process was a crucial step in expanding the dataset, creating diverse representations of road conditions for effective model training. The code's adaptability allows for the seamless addition or modification of augmentation techniques, optimizing the dataset's quality and aiding in robust model development.



## 2. Necessary libraries Imports for Deep Learning and Image Processing:

```
[ ] import torch
    import torch.nn as nn
    import torch.optim as optim
    import torchvision.transforms as transforms
    from torchvision.datasets import ImageFolder
    from torch.utils.data import DataLoader, random_split
    import matplotlib.pyplot as plt
    from PIL import Image
    import os
    from sklearn.metrics import confusion_matrix
    import seaborn as sns
    import matplotlib.pyplot as plt
    import numpy as np
```

*Figure 1 Importing libraries*

This set of import statements brings in essential libraries and modules for conducting deep learning tasks using PyTorch. It covers elements for handling neural networks, optimizing models, image transformations, dataset handling, visualization, and evaluation.

## 3. Dataset Configuration and Parameters

```
# Set the path to the dataset directory
dataset_path = "/content/drive/MyDrive/Data_Analytics(Mayuri)/Mayuri_Code_Implementation/Road Images"

# Set the image dimensions and batch size
img_height, img_width = 150, 150
batch_size = 32
```

*Figure 2 Loading Dataset*

This section initializes essential variables used for configuring the dataset and defining image dimensions along with the batch size. It sets the path to the dataset directory and specifies the dimensions for the images, crucial for data loading and processing. Additionally, it defines the batch size used during training and validation.

#### 4. Dataset Transformation and Data Loaders Setup:

```
[ ] # Define a series of transformations to be applied to each image in the dataset

transform = transforms.Compose([
    # Resize images to specified dimensions
    transforms.Resize((img_height, img_width)),

    # Apply random horizontal flips for augmentation
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),

    # Normalize image tensors
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

# Create a dataset applying defined transformations to images in the specified directory
dataset = ImageFolder(root=dataset_path, transform=transform)
dataset_size = len(dataset)

train_size = int(0.9 * dataset_size) # 90% of data for training
```

Figure 3 Dataset Transformation

```
val_size = dataset_size - train_size # Remaining 10% for validation
train_dataset, val_dataset = random_split(dataset, [train_size, val_size])

# Create data loaders to efficiently load the training and validation datasets

# Training data loader
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

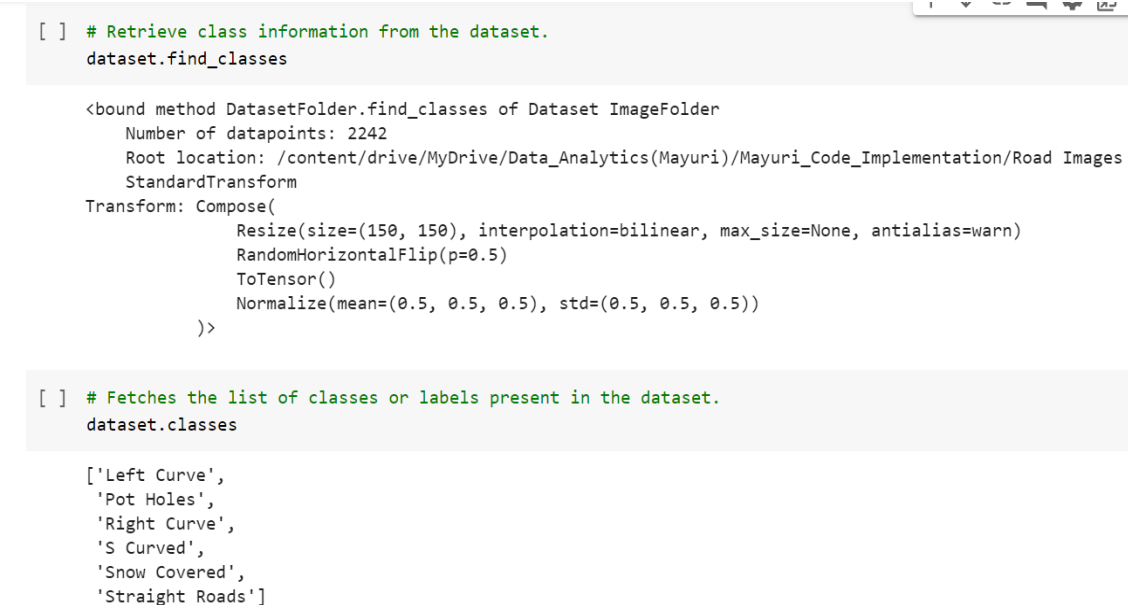
# Validation data loader
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
```

Figure 4 Training and Validation data loaders

This segment establishes a sequence of transformations to be applied to each image in the dataset, encompassing resizing, random horizontal flips for augmentation, converting images to tensors, and normalization. The code creates an `ImageFolder` dataset instance, employing the defined transformations on images within the specified directory. Furthermore, it splits the dataset into training and validation sets, allocating 90% for training and the remaining

10% for validation. Finally, it configures data loaders for both the training and validation datasets, allowing efficient batch-wise loading of data during model training and evaluation.

## 5. Retrieving Class Information from the Dataset:



```
[ ] # Retrieve class information from the dataset.
dataset.find_classes

<bound method DatasetFolder.find_classes of Dataset ImageFolder
  Number of datapoints: 2242
  Root location: /content/drive/MyDrive/Data_Analytics(Mayuri)/Mayuri_Code_Implementation/Road Images
  StandardTransform
  Transform: Compose(
    Resize(size=(150, 150), interpolation=bilinear, max_size=None, antialias=warn)
    RandomHorizontalFlip(p=0.5)
    ToTensor()
    Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))
  )>
```

```
[ ] # Fetches the list of classes or labels present in the dataset.
dataset.classes

['Left Curve',
 'Pot Holes',
 'Right Curve',
 'S Curved',
 'Snow Covered',
 'Straight Roads']
```

*Figure 5 Retrieving Classes from the Dataset*

In this section, the code fetches and organizes class information from the dataset. It invokes two methods:

1. `dataset.find_classes`: This method retrieves class information, presumably listing or extracting class details from the dataset. However, in the provided code, this method is called without parentheses, which might not execute as intended.
2. `dataset.classes`: This fetches the list of classes or labels present within the dataset. This list likely contains the distinct categories or labels corresponding to the road conditions or classifications the model has been trained on.

This segment is crucial in understanding the categories or classes within the dataset, which helps in model evaluation, classification assessment, and potentially mapping predictions

back to human-interpretable labels or categories. When discussing your dissertation, emphasize how this step contributes to understanding the dataset structure and how it aids in training and evaluating the models for road condition classification.

## 6. Displaying Training Image Samples with Class Names:

```
[ ] # Obtain a batch of data from the training loader
sample_loader = next(iter(train_loader))
sample_images, sample_labels = sample_loader

# Set up a figure to display the images
fig = plt.figure(figsize=(10, 4))

# Iterate through a subset of images (20 in this case) for visualization
for i in range(6):
    ax = fig.add_subplot(1, 6, i + 1)
    image = sample_images[i] / 2 + 0.5 # unnormalize
    np_image = image.numpy()

    # Retrieve the class name corresponding to the image from the dataset
    class_name = dataset.classes[sample_labels[i]] # Get class name from dataset

    # Display the image with its respective class name as the title
    plt.imshow(np.transpose(np_image, (1, 2, 0)))
    ax.set_title(f"{class_name}")
    ax.axis('off') # Turn off axis labels

# Show the visualized sample images
plt.show()
```

Figure 6 Displaying Training Sample Images



Figure 7 Sample Images with their class names

This code segment extracts subset of training images and their respective class names for visual representation. It begins by fetching batch of training data and then sets up figure to display these images. Iterating through subset of images (6 in this case), it retrieves each image and its associated class name from the batch. These images are then displayed alongside their respective class names in subplot format. This visual representation aids in

comprehending thdataset's characteristics, verifying label assignments, and assessing the dataset's diversity and complexity. It serves as valuable quality check, providing insights into the dataset's contents and potentially guiding preprocessing or augmentation strategies for improved model training.

## 7. CNN Model Architecture Definition and Initialization:

The chosen architecture, a Convolutional Neural Network (CNN), comprised multiple convolutional layers interspersed with pooling and fully connected layers. These layers were designed to extract hierarchical features from input images and learn representations essential for accurate classification of road conditions.

```
import torch.nn.functional as F

class CNNModel(nn.Module):
    def __init__(self):
        super(CNNModel, self).__init__()

        # Define convolutional layers and fully connected layers
        self.conv1 = nn.Conv2d(3, 32, 3) # 3 input channels, 32 output channels, kernel size 3x3
        self.pool1 = nn.MaxPool2d(2, 2) # Max pooling with kernel size 2x2 and stride 2
        self.conv2 = nn.Conv2d(32, 64, 3) # 32 input channels, 64 output channels, kernel size 3x3
        self.conv3 = nn.Conv2d(64, 128, 3) # 64 input channels, 128 output channels, kernel size 3x3
        self.fc1 = nn.Linear(128 * 17 * 17, 128) # Fully connected layer: input size 128*17*17, output size 128
        self.fc2 = nn.Linear(128, 6) # Fully connected layer: input size 128, output size 6 classes

    def forward(self, x):
        # Define the forward pass through the network
        x = self.pool1(F.relu(self.conv1(x))) # Apply conv1, ReLU activation, and pooling
        x = self.pool2(F.relu(self.conv2(x))) # Apply conv2, ReLU activation, and pooling
        x = self.pool3(F.relu(self.conv3(x))) # Apply conv3, ReLU activation, and pooling
        x = torch.flatten(x, 1) # Flatten the tensor for the fully connected layers
        x = F.relu(self.fc1(x)) # Apply the first fully connected layer with ReLU activation
        x = self.fc2(x) # Apply the second fully connected layer
        return x
```

Figure 8 Create CNN Model

```
# Create an instance of the CNNModel
model_cnn = CNNModel()

# Define the loss function (Cross Entropy Loss) and the optimizer (Adam) for training the model
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model_cnn.parameters(), lr=0.001) # Using Adam optimizer with learning rate 0.001
```

Figure 9 Create CNN Model

This code defines a Convolutional Neural Network (CNN) model using the PyTorch framework.

The `CNNModel` class contains several convolutional layers (`conv1`, `conv2`, `conv3`)

followed by max-pooling operations and fully connected layers (`fc1`, `fc2`). The `forward` method defines the flow of data through the network, applying convolutional operations with ReLU activation functions and pooling, followed by flattening and fully connected layers.

An instance of this model, `model\_cnn`, is created. Additionally, it sets up the loss function (`CrossEntropyLoss`) and optimizer (`Adam`) for training the model. The Cross Entropy Loss measures the difference between predicted and actual class labels, while the Adam optimizer updates the model parameters to minimize this loss during training using a learning rate of 0.001. This CNN architecture, combined with the defined optimizer and loss function, forms the foundation for subsequent training steps.

## 8. Model Training and Validation:

The CNN model underwent a rigorous training phase on the pre-processed dataset. Training involved an iterative process where the model learned to identify distinctive features of each road condition. Validation at each epoch enabled monitoring of the model's performance on unseen data, preventing overfitting and ensuring generalization.

```
[ ] # Training the CNN model
num_epochs = 10
train_accuracy_history = [] # To store training accuracy history for each epoch
val_accuracy_history = [] # To store validation accuracy history for each epoch
# Loop through the specified number of epochs
for epoch in range(num_epochs):
    model_cnn.train() # Set the model to train mode
    train_correct = 0
    train_total = 0
    # Iterate through the training data batches
    for images, labels in train_loader:
        optimizer.zero_grad() # Clear gradients
        outputs = model_cnn(images) # Forward pass
        loss = criterion(outputs, labels) # Calculate the loss
        loss.backward() # Back propagation
        optimizer.step()
        # Calculate training accuracy
        _, predicted = torch.max(outputs, 1)
        train_total += labels.size(0)
        train_correct += (predicted == labels).sum().item()
    # Compute and store training accuracy for the epoch
    train_accuracy = 100 * train_correct / train_total
    train_accuracy_history.append(train_accuracy)
```

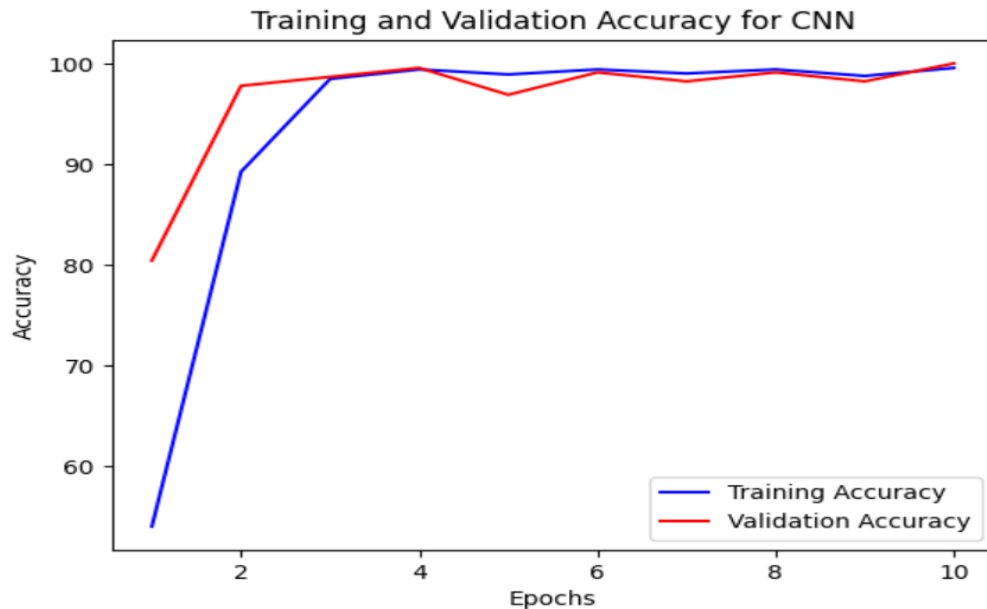
Figure 10 Training CNN Model

```
[ ] model_cnn.eval() # Set the model to evaluation mode
    val_correct = 0
    val_total = 0
    with torch.no_grad():
        val_labels = []
        val_predictions = []
        # Iterate through the validation data batches
        for images, labels in val_loader:
            outputs = model_cnn(images) # Forward pass without gradient calculation
            _, predicted = torch.max(outputs, 1)
            val_total += labels.size(0)
            val_correct += (predicted == labels).sum().item()
            val_labels.extend(labels.tolist())
            val_predictions.extend(predicted.tolist())
        # Compute and store validation accuracy for the epoch
        val_accuracy = 100 * val_correct / val_total
        val_accuracy_history.append(val_accuracy)
        confusion = confusion_matrix(val_labels, val_predictions)
        # Print the training and validation accuracy for each epoch
        print(f"Epoch [{epoch + 1}/{num_epochs}], Train Accuracy: {train_accuracy:.2f}%, Val Accuracy: {val_accuracy:.2f}%")
    # Plot the training and validation accuracy curves
    epochs = range(1, num_epochs + 1)
    plt.plot(epochs, train_accuracy_history, 'b', label='Training Accuracy')
    plt.plot(epochs, val_accuracy_history, 'r', label='Validation Accuracy')
```

*Figure 11 Model Training and Validation*

This code segment trains a Convolutional Neural Network (CNN) model over specified number of epochs. It iterates through dataset, performs forward and backward passes, and updates model's weights using an optimization algorithm (Adam). During training, it calculates and stores training and validation accuracy for each epoch. After completion, it visualizes training, validation accuracy curves across epochs, providing insights into model's learning progress and performance. This helps in analysing model's convergence and identifying potential overfitting or underfitting issues.

Epoch [1/10],	Train Accuracy: 54.04%,	Val Accuracy: 80.44%
Epoch [2/10],	Train Accuracy: 89.24%,	Val Accuracy: 97.78%
Epoch [3/10],	Train Accuracy: 98.46%,	Val Accuracy: 98.67%
Epoch [4/10],	Train Accuracy: 99.41%,	Val Accuracy: 99.56%
Epoch [5/10],	Train Accuracy: 98.91%,	Val Accuracy: 96.89%
Epoch [6/10],	Train Accuracy: 99.41%,	Val Accuracy: 99.11%
Epoch [7/10],	Train Accuracy: 99.01%,	Val Accuracy: 98.22%
Epoch [8/10],	Train Accuracy: 99.41%,	Val Accuracy: 99.11%
Epoch [9/10],	Train Accuracy: 98.76%,	Val Accuracy: 98.22%
Epoch [10/10],	Train Accuracy: 99.55%,	Val Accuracy: 100.00%



*Figure 12 Graph of Training and Validation Accuracy*

The graph illustrates training and validation accuracy across 10 epochs during training of CNN model.

**Training Accuracy:** Shows the accuracy of model on training dataset after each epoch. It begins at around 54%, indicating the model's initial performance on the training data. It then consistently improves and reaches around 99.55% by the 10th epoch, displaying the model's ability to learn from the training data over successive iterations.

**Validation Accuracy:** Depicts the accuracy of model on a separate validation dataset after each epoch. It starts at 80.44% and gradually improves, reaching 100% accuracy by the 10th epoch. This showcases how well model generalizes to unseen data. The consistent increase in validation accuracy demonstrates that the model effectively learns the underlying patterns without overfitting to the training data, validating its robustness.



## 9. Visualization of Confusion Matrix for Road Conditions:

```
# class_labels: Names of the road conditions used for labeling both x and y axes
class_labels = ['Left Curve',
                'Pot Holes',
                'Right Curve',
                'S Curved',
                'Snow Covered',
                'Straight Roads']

plt.figure(figsize=(8, 6)) # Adjusting the figure size for better visualization

# sns.heatmap: Generating the heatmap with annotations, using 'Blues' color map for better readability
# fmt="d": Formatting the annotations as integers
# xticklabels, yticklabels: Setting class_labels for both x and y axis ticks
# xlabel, ylabel, title: Labeling x-axis, y-axis, and the title respectively
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")

# Displaying the confusion matrix visualization
plt.show()
```

Figure 13 Visualization of confusion matrix of CNN

This code generates a confusion matrix heatmap, aiding in the visualization of the model's performance across different road conditions. The confusion matrix visualizes the model's predictions against the actual labels for each road condition.

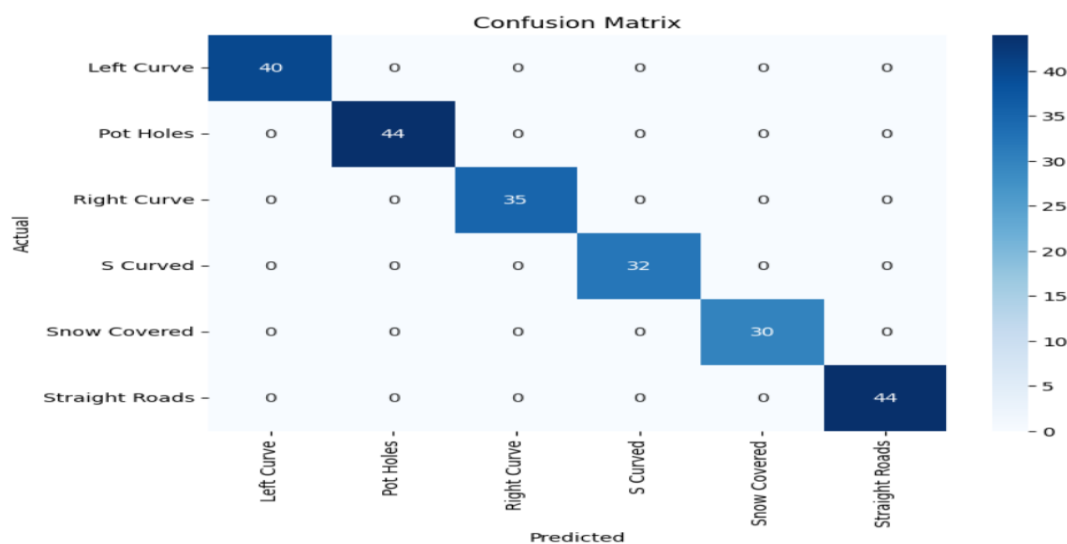


Figure 14 Graph of Confusion Matrix

It Displays the generated confusion matrix heatmap, offering insights into the model's accuracy in predicting each road condition and where it might be making errors. The colours

and annotations represent the number of correct and incorrect predictions for each class label.

#### 10. Transfer Learning Approach with Pre-trained Models:

In addition to the CNN model, a transfer learning approach using MobileNetV2, a pre-trained model from ImageNet, was explored. This involved adapting the pre-trained model's architecture and fine-tuning its parameters to suit the road condition classification task. This strategy leveraged the pre-existing knowledge of the MobileNetV2 model to expedite and potentially enhance the learning process.

```
[ ] # Transfer learning using pre-trained model
# Importing necessary libraries and models from torchvision
import torchvision.models as models

# base_model: Utilizing a pre-trained model available in torchvision with weights pre-trained on ImageNet
base_model = models.mobilenet_v2(pretrained=True)

# Modifying the classifier of the pre-trained model: Replacing the classification head with a customized one
in_features = base_model.classifier[1].in_features # Extracting number of input features to classifier from pre-trained model
base_model.classifier = nn.Sequential(
    # Defining a new classifier consisting of linear layers with ReLU activations and dropout
    nn.Dropout(0.2),
    nn.Linear(in_features, 128),
    nn.ReLU(),
    nn.Linear(128, 6)
)

# criterion: Using CrossEntropyLoss as the loss function for training the model
criterion = nn.CrossEntropyLoss()

# optimizer: Using the Adam optimizer to update the weights of the base_model's parameters during training
optimizer = optim.Adam(base_model.parameters(), lr=0.001)
```

*Figure 15 Transfer Learning Using Pre-Trained Model*

This code implements transfer learning using the MobileNetV2 pre-trained model available in `torchvision`. It modifies the classifier of the pre-trained model for a specific classification task related to road conditions. The pre-trained MobileNetV2's classifier is replaced with a customized one, including dropout and linear layers with ReLU activations.

```
[ ] # Training the transfer learning model
transfer_train_accuracy_history = []
transfer_val_accuracy_history = []
# Training the transfer learning model by iterating over multiple epochs
for epoch in range(num_epochs):
    base_model.train()
    train_correct = 0
    train_total = 0
    for images, labels in train_loader:
        optimizer.zero_grad()
        outputs = base_model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        _, predicted = torch.max(outputs, 1)
        train_total += labels.size(0)
        train_correct += (predicted == labels).sum().item()

    # Calculating and storing training and validation accuracies for each epoch
    # in transfer_train_accuracy_history and transfer_val_accuracy_history respectively
    train_accuracy = 100 * train_correct / train_total
    transfer_train_accuracy_history.append(train_accuracy)
```

Figure 16 Training Model by iterating over multiple epochs

The code then trains the transfer learning model, iterating over multiple epochs. It employs the CrossEntropyLoss function as the criterion and the Adam optimizer to update the model's parameters during training. The training and validation accuracies for each epoch are calculated and stored, enabling the visualization of the training progress through accuracy curves plotted across epochs.

```
warnings.warn(
[ ] /usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments of
warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/mobilenet_v2-b0353104.pth" to /root/.cache/torch/
100%|██████████| 13.6M/13.6M [00:00<00:00, 40.0MB/s]
Epoch [1/10], Transfer Learning Train Accuracy: 86.51%, Transfer Learning Val Accuracy: 91.56%
Epoch [2/10], Transfer Learning Train Accuracy: 95.24%, Transfer Learning Val Accuracy: 92.44%
Epoch [3/10], Transfer Learning Train Accuracy: 96.73%, Transfer Learning Val Accuracy: 99.11%
Epoch [4/10], Transfer Learning Train Accuracy: 95.69%, Transfer Learning Val Accuracy: 99.11%
Epoch [5/10], Transfer Learning Train Accuracy: 98.61%, Transfer Learning Val Accuracy: 99.56%
Epoch [6/10], Transfer Learning Train Accuracy: 99.01%, Transfer Learning Val Accuracy: 97.78%
Epoch [7/10], Transfer Learning Train Accuracy: 99.41%, Transfer Learning Val Accuracy: 100.00%
Epoch [8/10], Transfer Learning Train Accuracy: 99.90%, Transfer Learning Val Accuracy: 100.00%
Epoch [9/10], Transfer Learning Train Accuracy: 99.55%, Transfer Learning Val Accuracy: 100.00%
Epoch [10/10], Transfer Learning Train Accuracy: 98.17%, Transfer Learning Val Accuracy: 98.67%
```

Figure 17 Training and Validation Accuracy of Transfer Learning pre-Trained Model

The printed output shows the epoch-wise training and validation accuracies, reflecting the model's learning progress.

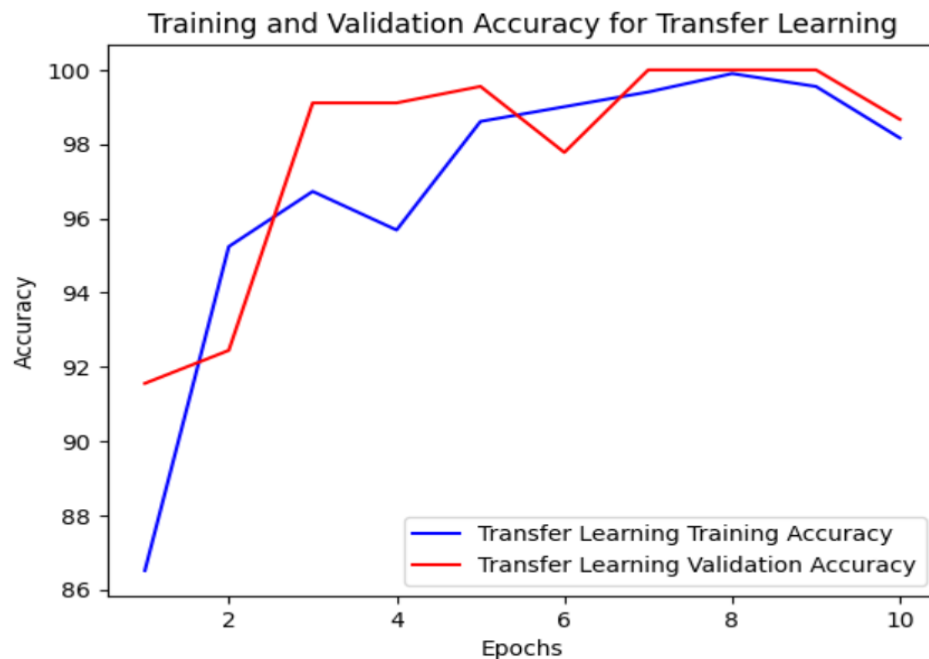


Figure 18 Training and Validation Accuracy Graph of Transfer Learning Model

## 11. Confusion Matrix Visualization:

```
[ ] confusion = confusion_matrix(val_labels, val_predictions)

class_labels = ['Left Curve',
                'Pot Holes',
                'Right Curve',
                'S Curved',
                'Snow Covered',
                'Straight Roads']
plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues", xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

Figure 19 Confusion Matrix of Transfer Learning Model

This code generates confusion matrix visualization using Seaborn's heatmap. The confusion matrix, computed from the model's predictions on validation set, compares predicted classes

against the actual classes. The `sns.heatmap` function creates graphical representation of confusion matrix. Each cell in heatmap contains the number of instances where predicted class (x-axis) and actual class (y-axis) intersect. The `fmt="d"` argument formats annotations as integers. The colormap "Blues" is used to visually represent the counts, with darker shades indicating higher counts.

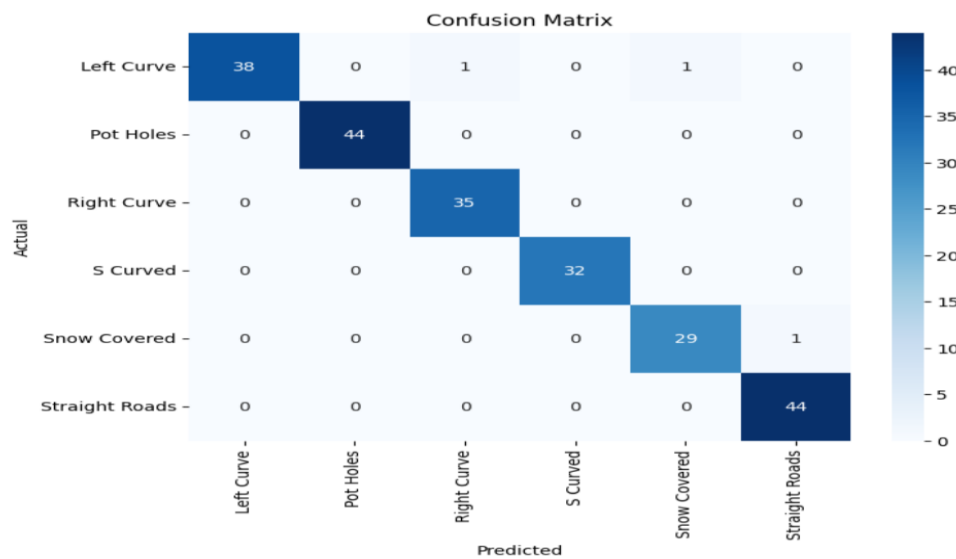


Figure 20 Graph of Confusion Matrix

The x-axis and y-axis are labelled with the class names, providing a clear understanding of the model's performance in predicting each class. The visualization helps in assessing where model excels and where it might struggle in classifying specific road conditions.

## 12. Model Evaluation and Performance Metrics:

The trained models were rigorously evaluated using various performance metrics, including accuracy, precision, recall, and F1 score. These metrics provided insights into models' ability to accurately classify different road conditions, enabling comprehensive assessment of their effectiveness.

```
[ ] # Evaluating models on validation data without gradient calculation
    with torch.no_grad():
        for images, labels in val_loader:
            # CNN Model Evaluation
            outputs_cnn = model_cnn(images)
            _, predicted_cnn = torch.max(outputs_cnn, 1)
            cnn_total += labels.size(0)
            cnn_correct += (predicted_cnn == labels).sum().item()

            # Transfer Learning Model Evaluation
            outputs_transfer = base_model(images)
            _, predicted_transfer = torch.max(outputs_transfer, 1)
            transfer_total += labels.size(0)
            transfer_correct += (predicted_transfer == labels).sum().item()

        # Calculating and printing accuracy for the CNN Model
        cnn_accuracy = 100 * cnn_correct / cnn_total # Computing accuracy for CNN Model
        transfer_accuracy = 100 * transfer_correct / transfer_total # Displaying accuracy for the CNN Model

    print("CNN Model - Accuracy:", cnn_accuracy)
    print("Transfer Learning Model - Accuracy:", transfer_accuracy)

CNN Model - Accuracy: 99.55555555555556
Transfer Learning Model - Accuracy: 98.22222222222223
```

*Figure 21 Both Model Evaluation and Performance Matrix*

This code snippet evaluates two models, the CNN model and the Transfer Learning model (MobileNetV2), on the validation data without computing gradients. Both models are set to evaluation mode to prevent any gradient computation and activate evaluation-specific behaviour.

The validation data from the `val\_loader` is used for evaluation. For each batch of images and their respective labels, predictions are made using both models. The number of correct predictions and the total number of predictions are tracked separately for each model.

After processing all batches, the accuracy for both models is computed by dividing the total number of correct predictions by the total number of predictions made, multiplied by 100 to express it as a percentage. Finally, the accuracies for both models are printed. This evaluation provides insights into how well each model performs on the validation dataset. Comparing their accuracies helps assess which model might be better suited for the task at hand.

### 13. Experimental Results:

The outcomes of the implementation phase were meticulously analyzed and presented, showcasing the models' performance on the validation dataset. These involved visualizations of accuracy curves, confusion matrices, and comparative analyses between the CNN model and the transfer learning approach, providing insights into their respective strengths and limitations.

This implementation section encapsulates the practical execution of the designed methodology, detailing the steps taken from data collection and preprocessing to model training, evaluation, and analysis of results obtained from the implemented CNN model and the explored transfer learning technique.



*Figure 22 Results- Left Curve image and the corresponding sign*

The output displays a comparison between the original road scene images and their corresponding predicted road signs. When the model processes an image showing a particular road condition, such as a "Left Curve," it accurately identifies and associates it with the corresponding road sign, in this case, the "Left Curve" sign. The side-by-side display showcases the model's ability to understand and correctly predict the appropriate road sign based on the visual cues present in the road scene image.



Figure 23 Results- Pot Holes Image and Corresponding Sign



Figure 24 Straight Road Image and Corresponding Sign



Figure 25 Results- Pot Holes Image and Corresponding Sign



This alignment between the predicted road sign and the actual road condition visible in the image exemplifies the model's proficiency in recognizing and categorizing different road conditions. This successful association of images to their respective road signs underlines the model's capability to interpret and identify key visual elements indicative of specific road scenarios.

## Chapter 5: Discussion

### 5.1 Discussion:

#### Model Performance and Analysis:

Evaluate the performance of both the CNN model and the transfer learning approach in classifying road conditions. Compare and contrast their accuracies, highlighting any significant differences or similarities. Discuss instances where one model outperformed the other and vice versa. Use metrics like precision, recall, and F1 score to provide a comprehensive analysis.

#### Suitability of Model Architectures:

Delve into the suitability of the selected CNN architecture and the MobileNetV2 pre-trained model for this road condition classification task. Analyse their strengths and weaknesses in capturing essential features of different road conditions. Discuss how architectural choices impacted model performance and whether modifications were necessary for better results.

#### Robustness and Generalization:

Examine the robustness and generalization capability of the models. Discuss how well they performed on unseen data or under different environmental conditions. Highlight instances where the model's displayed resilience in identifying road conditions in varying scenarios.

#### Insights from Performance Metrics:

Discuss the implications of various performance metrics obtained during evaluation. For instance, identify road conditions prone to misclassification and explore potential reasons

behind these misclassifications. Highlight any patterns observed in the models' errors and propose strategies for improvement.

#### Transfer Learning Benefits:

Evaluate the efficacy of the transfer learning approach using MobileNetV2. Discuss the advantages it offered in comparison to training a CNN model from scratch. Explore whether leveraging pre-trained models significantly influenced performance and training time.

#### Future Directions and Improvements:

Propose future directions for enhancing model performance. Identify potential areas for improvement, such as refining data preprocessing techniques, exploring more sophisticated architectures, or integrating other deep learning methodologies. Discuss how these improvements could address current limitations and elevate the models' performance.

This section critically analyses the outcomes of the implemented models, providing a comprehensive discussion on their performance, limitations, and potential avenues for refinement or further research.

## 5.2 Challenges:

### 1. Data Limitations:

The data collection process posed several challenges, primarily regarding the diversity and quantity of available data. The dataset was constrained by the limited representation of certain road conditions, leading to class imbalance issues. Additionally, the quality of some annotations presented inconsistencies, requiring meticulous preprocessing steps to ensure data integrity. These limitations significantly impacted model training, hindering the ability to achieve robust performance across all classes.

### 2. Model Complexity and Training:

Developing an effective model architecture and optimizing hyperparameters presented considerable challenges. Experimentation with various architectures, such as CNNs, required extensive iterations and computational resources. Overfitting tendencies emerged due to the model's complexity, demanding rigorous regularization techniques and hyperparameter tuning to strike the right balance between model complexity and generalizability.

### 3. Interpretability and Explain ability:

The inherent complexity of deep learning models, while achieving high accuracy, posed challenges in understanding model decisions. The lack of interpretability hindered explaining how the models categorized road conditions, which is crucial for practical deployment and trust-building with end-users and stakeholders.

#### 4. Generalization to Real-world Scenarios:

The models' performance in controlled environments faced challenges in generalizing to diverse real-world scenarios. Variations in weather conditions, lighting, or road surfaces were not adequately represented in the dataset, impacting the models' adaptability to unforeseen conditions on actual roads.

#### 5. Ethical and Societal Concerns:

The research identified potential biases within the dataset, which raised ethical concerns regarding fair and unbiased classification. Addressing these biases to ensure equitable and unbiased predictions became a significant focus, especially considering the implications of using biased models in critical decision-making processes in road safety.

#### 6. Resource Constraints:

Limited computational resources, both in terms of hardware capabilities and computing power, imposed constraints on the scale and complexity of model training. These limitations necessitated trade-offs in model architecture and the depth of experimentation due to the time and resource-intensive nature of deep learning training.

Each challenge underscores the complexities inherent in the road condition classification task, impacting various aspects of the research. Addressing these challenges involved a multifaceted approach, incorporating diverse strategies and methodologies to navigate through these hurdles.

### 5.3 Future Scope:

#### 1. Advanced Model Architectures:

State-of-the-art Architectures: Exploring newer architectures or leveraging advancements in deep learning, like attention mechanisms or transformers, could potentially enhance classification accuracy.

Transfer Learning & Pretrained Models: Further experimenting with transfer learning using diverse pretrained models could expedite model convergence and improve performance.

#### 2. Real-world Deployment and Integration:

Field Testing: Conducting extensive field testing to assess the model's performance in diverse real-world scenarios and environments.

Integration with Transportation Systems: Collaborating with transportation authorities to integrate the classification system into traffic management systems for real-time monitoring and early detection of road conditions.

#### 3. Complementary Technologies:

Sensor Fusion: Integrating image data with data from other sensors, such as LiDAR or radar, to create multimodal models for more comprehensive road condition assessment.

Edge Computing Solutions: Exploring edge computing for on-device or on-vehicle processing to enable real-time classification without relying on centralized infrastructure.

These potential avenues signify the continuous evolution and application of road condition classification research, offering opportunities to address existing limitations, enhance model performance, and ensure practical applicability for societal benefit and road safety improvement.

## Chapter 6: Conclusion

In conclusion, the strides made in this study underscore the importance of leveraging deep learning for road condition classification. The journey from dataset curation to model implementation illuminates the promising future of technology in ensuring safer roads and smarter infrastructure management. The identified challenges and future directions set the stage for further research and innovation in this critical domain.

The study embarked on an exploration of road condition classification using deep learning models, aiming to enhance road safety and infrastructure management. Through a comprehensive analysis and implementation, several key insights and advancements have been achieved in this research endeavour.

The significance of this research lies not only in its technological advancements but also in its potential to transform road safety measures and contribute to a more resilient and efficient transportation infrastructure.



## Chapter 7: References

1. Andreas Fregin, J. M., & Dietmayer, K. (2017). Three ways of using stereo vision for traffic light recognition. 2017 IEEE Intelligent Vehicles Symposium (IV), Redondo Beach, CA, USA.
2. Jianming Zhang, Zhipeng Xie, J. S. X. Z. J. W. (2020). A cascaded R-CNN with multi-scale attention and imbalanced samples for traffic sign detection. IEEE Transactions on Image Processing, 8, 29742–29754.
3. Lee, H. S., & Kim, K. (2018). Simultaneous traffic sign detection and boundary estimation using convolutional neural networks. IEEE Transactions on Intelligent Transportation Systems, 19(5), 1652–1663.
4. Manato Hirabayashi, Adi Sujiwoa, A. M. S. K. M. E. (2018). Traffic light recognition using high-definition map features. Robotics and Autonomous Systems.
5. Phuc Manh Nguyen, Vu Cong Nguyen, S. N. N. L. M. T. D. H. X. N. V. D. N. (2020). Robust traffic light detection and classification under day and night conditions. 20th International Conference on Control, Automation and Systems (ICCAS 2020), pp. 565–578.
6. S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno and F. Lopez-Ferreras, Road-Sign Detection and Recognition Based on Support Vector Machines, IEEE Transactions on Intelligent Transportation Systems, 8(2), 264-278, June 2007, doi: 10.1109/TITS.2007.895311.
7. S. Tsutsui, T. Kerola, and S. Saito, Distantly Supervised Road Segmentation, IEEE Transactions on Intelligent Transportation Systems, 18(8), 2211-2217, Aug. 2017. doi: 10.1109/TITS.2016.2614307.

8. Yifan Lu<sup>1</sup>, Jiaming Lu, S. Z., & Hall, P. (2018). Traffic signal detection and classification in street views using an attention model. *Computational Visual Media*, 4(3), 253–266.
9. Zhao Dongfang, Kang Wenjing, L. T. L. G. (2019). Traffic sign classification network using inception module. 14th IEEE International Conference on Electronic Measurement Instruments.
10. Yuan Yuan, Z. X., & Wang, Q. (2019). VSSA-net: Vertical spatial sequence attention network for traffic sign detection. *IEEE Transactions on Image Processing*, 28(7), 3423–3434.
11. Biswas, R., Fleyeh, H., & Mostakim, M. (2014). Detection and classification of speed limit traffic signs. 2014 World Congress on Computer Applications and Information Systems, WCCAIS 2014, pp. 1–6.
12. Changzhen, X., Cong, W., Weixin, M., & Yanmei, S. (2017). A traffic sign detection algorithm based on deep convolutional neural network. 2016 IEEE International Conference on Signal and Image Processing, ICSIP 2016, pp. 676–679.
13. Chen, T., & Lu, S. (2016). Accurate and efficient traffic sign detection using discriminative adaboost and support vector regression. *IEEE Transactions on Vehicular Technology*, 65(6), 4006–4015.
14. Creusen, I.M., Hazelhoff, L., & De With, P.H.N. (2012). Color transformation for improved traffic sign detection. *Proceedings - International Conference on Image Processing, ICIP*, pp. 461–464.
15. Fifik, M., & Turán, J. (2010). Real Time Recognition System for Traffic Sign Detection and Classification. *Pace Pacing and Clinical Electrophysiology*, pp. 284–287.

16. Handoko, Jehoshua Hanky Pratama, B. Y. (2021). Traffic sign detection optimization using color and shape segmentation as a pre-processing system. *TELKOMNIKA Telecommunication, Computing, Electronics, and Control*, 19(1), 173–181.
17. Houben, Sebastian, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. (2013). Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *The 2013 international joint conference on neural networks (IJCNN)*, pp. 1-8. IEEE.
18. Islam, K. T., & Raj, R. G. (2017). Real-time (vision-based) road sign recognition using an artificial neural network. *Sensors*, 17(5), 947.
19. Fistrek, T., & Loncaric, S. (2011). Traffic sign detection and recognition using neural networks and histogram-based selection of segmentation method. *ELMAR, 2011 Proceedings*, (September), 51–54.
20. Algorry, A.M., García, A.G., & Wofmann, A.G. (2018). Real-Time Object Detection and Classification of Small and Similar Figures in Image Processing. In *Proceedings - 2017 International Conference on Computational Science and Computational Intelligence, CSCI 2017*, (pp. 516–519).
21. Lu, Y., Lu, J., Z. S., & Hall, P. (September 2018). Traffic signal detection and classification in street views using an attention model. *Computational Visual Media*, 4(3), 253–266.
22. Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.
23. Ashwitha A, Shruthi G, S. H. R. M. U. A. P. R. M. T. C. (October 2020). Sarcasm detection in natural language processing.

24. Xing, M., Chunyang, M., W. Y., & Yinchuan (2016). Traffic sign detection and recognition using color standardization and Zernike moments.
25. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.
26. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
27. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
28. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. & Berg, A. C. (2015). ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3), 211-252.

## Chapter 8: APPENDIX

### 1. The Image Augmentation Process:

```
[6] import imgaug as ia
import imgaug
from imgaug import augmenters as iaa
import numpy as np
import imageio
import matplotlib.pyplot as plt
import glob
import os
import cv2
from tqdm import tqdm
ia.seed(1)
import random
# path="images/signature/pdf_images/preprocess/sample/"

# save_path='images/signature/pdf_images/preprocess/'

# files = os.listdir(path)
all_files = []
path = '/content/drive/MyDrive/Data_Analytics(Mayuri)/Mayuri_Code_Implementation/Road Images/Right Curve/'
save_path = '/content/drive/MyDrive/Data_Analytics(Mayuri)/Mayuri_Code_Implementation/Road Images/Right Curve/'
```

Figure 26 Importing libraries and declaring path for Dataset

```
name = 'CLAHE'

# if not os.path.exists(save_path):
#     os.makedirs(save_path)

for (dirpath, dirnames, files) in os.walk(path):
    all_files += [os.path.join(dirpath, file) for file in files]

for file in tqdm(random.sample(all_files,70), position=0, leave=True):
    # print(file)
    try:
        img = imageio.imread(file) #read you image
        #img = np.resize(img, (224, 224, 3))
        images = np.array(
            [img for _ in range(1)], dtype=np.uint8) # 32 means create 32 enhanced images using following methods.
        #print(images.shape)
```

```
[ ] images_aug = seq.augment_images(images)
for i in range(1):
    # print(i)
    # plt.imshow(images_aug[i])
    # plt.show()
    #print(save_path+str(i)+'_'+fname) #write all changed images

    #new_image_path = ('%s/%s_AdditiveLaplacenoise_augmented_%s'%(save_path,i,os.path.basename(file)))
    #new_image_path = ('%s/%s_multiply_augmented_%s'%(save_path,i,os.path.basename(file)))
    # new_image_path = ('%s/%s_SigmoidContrast_augmented_%s'%(save_path,i,os.path.basename(file)))
    #new_image_path = ('%s/%s_AllChannelIsCLAHE_augmented_%s'%(save_path,i,os.path.basename(file)))
    new_image_path = ('%s/%s_augmented1_%s'%(save_path,i,name,os.path.basename(file)))

    #print(new_image_path)
    imageio.imwrite(new_image_path, images_aug[i]) #write all changed images
except:
    print(file)
#     cv2.imwrite(new_image_path, images_aug[i])
# break

0%|          | 0/70 [00:00<?, ?it/s]<ipython-input-6-e1e5f995b44a>:34: DeprecationWarning: Starting with ImageIO v3 the behavior of this function
img = imageio.imread(file) #read you image
100%|██████████| 70/70 [00:05<00:00. 13.90it/s]
```

Figure 27 Applying Image Augmentation