

**DECLARATION**

I QUADRI IBRAHIM ADETOLA with Matriculation Number 22/10977, from the department of CYBERSECURITY, hereby declare that this project titled: AI-BASED DETECTION OF ADVERSARIAL IMAGE ATTACKS ON AUTONOMOUS SYSTEMS ( A CASE STUDY ON: SELF-DRIVEN VEHICLES) is my original work and has not been submitted previously in any part or whole for the award of any degree, diploma, or certificate in any institution. All materials used from other sources have been duly acknowledged.

This project was completed under the supervision of PROF. M.K. AREGBESHOLA

I take full responsibility for the contents of this work.

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CERTIFICATION**

This is to certify that the project title “AI-Based Detection of Adversarial image attacks on autonomous systems (A case study on: autonomous vehicles)” was carried out by QUADRI IBRAHIM.A. with matric number 22/10977 form the department of Cybersecurity, College of Pure and Applied Sciences, Caleb University, Lagos State. The project work is considered adequate in partial fulfillment of the requirement to the award of a Bachelor of Science in Cybersecurity.

PROF. M.K. AREGBESHOLA ……………………

(PROJECT SUPERVISOR) SIGNATURE/DATE

DR. ADENIYI AKANNI …………………….

(H.O.D OF COMPUTER SCIENCE) SIGNATURE/DATE

**DEDICATION**

I humbly dedicate this project to GOD Almighty for giving me the strength, guidance, grace and divine protection towards the completion of the project. I also dedicate this project to my beloved parents, Mr. and Mrs. Quadri, my wonderful siblings, colleagues and distinguished lecturers who supported me through my journey.

**ACKNOWLWDGEMENT**

First and foremost, I would like to express my sincere gratitude to GOD Almighty for his guidance and blessings throughout my academic journey. I extend my heartfelt thanks to my project supervisor, Prof. M.K. AREGBESOLA for his dedication, commitment and guidance which has been instrumental in the successful completion of this project. My lovely parents as well, for their endless and relentless support and motivation throughout my program, I am grateful for your unwavering love and support and motivation during this endeavor.

Finally, CALEB UNIVERSITY for an experience well shared, from my HOD, my lecturers down to my colleagues who have impacted knowledge to me in one way or the other, I say a very big thank you.

**ABSTRACT**

In recent years, autonomous systems have become more prevalent in various industries, the need to protect them from potential attacks is paramount. One such threat is the adversarial image attack, where malicious actors manipulate images in a way that can deceive AI systems and cause them to make incorrect decisions. To combat this threat, researchers have developed AI-based detection methods that can identify and mitigate adversarial image attacks on autonomous systems. These detection systems utilize machine learning algorithms to analyze images and detect any anomalies or inconsistencies that may indicate a potential attack. Various domains have been influenced by the rapid growth of machine learning. Autonomous driving is an area that has tremendously developed in parallel with the advancement of machine learning. In autonomous vehicles, various machine learning components are used such as traffic lights recognition, traffic sign recognition, limiting speed and pathfinding. The integration of advanced image analysis using artificial intelligence (AI) is pivotal for the evolution of autonomous vehicles (AVs).

By incorporating these AI-based detection methods into autonomous systems, organizations can significantly enhance their security posture and ensure the integrity and reliability of their operations. As the threat landscape continues to evolve, it is imperative that organizations stay ahead of potential attacks and leverage the power of AI to protect their autonomous systems from malicious actors. This article provides a thorough review of the most significant datasets and the latest state-of-the-art AI solutions employed in image analysis for AVs. Datasets such as Cityscapes, NuScenes (motion sensor), and CARLA (Car learning to act) form benchmarks for training and evaluating different AI models, with unique characteristics catering to various aspects of autonomous driving. Key AI methodologies, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Transformer models, and Generative Adversarial Networks (GANs), are discussed. The article also presents a comparative analysis of various AI techniques in real-world scenarios, focusing on semantic image segmentation, 3D object detection, and vehicle control in virtual environments

However, these machine learning models are vulnerable to targeted sensor perturbations called adversarial attacks, which limit the performance of the applications. Therefore, implementing defense models against adversarial attacks has become an increasingly critical research area. The paper aims to summarize the latest adversarial attacks and defense models introduced in the field of autonomous driving with machine learning technologies as at 2025.

**TABLE OF CONTENT**

Title Page i

Declaration ii

Certification ii

Dedication iii

Acknowledgement iv

Abstract vi

Table of Content viii

List of Figures xiii

List of Tables xiv

List of Abbreviations xv

CHAPTER ONE: INTRODUCTION

1.1 Background of study 1

1.2 Definition of the problem 2

1.3 Significance of study 3

1.4 Aim and Objectives 4

1.4.1 Aim 4

1.4.2 Objective 5

1.5 Methodology 6

1.6 Scope of study and Limitation 8

1.6.1 Scope 8

1.6.2 Limitation 9

1.7 Future research trend 10

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction 13

2.2 Autonomous driving models 15

2.3 Adversarial attacks 16

2.4 Defending against adversarial attacks 19

2.5 Adversarial attacks method on autonomous systems 21

2.6 Adversarial robustness and strategies 24

2.6.1 Proactive 25

2.6.2 Reactive 25

2.7 Increasing the speed and efficiency of the defense models 27

CHAPTER THREE: METHODOLOGY

3.1Introduction 28

3.2 System Architecture 28

3.2.1 Input image processing module 30

3.2.2 Adversarial Perturbation detection module 30

3.2.3 Decision and alert module 31

3.3 Summary of the system architecture 32

3.4 Dataset acquisition and preprocessing 32

3.4.1 Data selection 32

3.4.2 Hugging face 33

3.4.3 Hugging face model *Dima806/traffic\_sign\_detection*  33

3.4.4 Data processing 34

3.5 Model selection (ViTs) and training 35

3.5.1 Vision Transformer (ViTs) selection 35

3.5.2 Transfer learning and fine-tuning 35

3.5.3 Training process (pipeline ) 36

3.6 Adversarial attack detection module 36

3.6.1 Model analysis and classification 37

3.6.2 Output and response handling 37

3.6.3 Integration with autonomous system (self-driven vehicle) 38

3.6.4 Performance consideration 38

3.6.5 Logging and feedback 39

3.7 Implementation environment and strategy 39

CHAPTER FOUR: DESIGN AND IMPLEMENTATION

4.1 Introduction 41

4.2 Development environment and tools 41

4.3 System architecture 42

4.3.1 User interface layer 43

4.3.2 Processing layer (application logic) 43

4.3.3 Inference layer (Backend ML integration) 43

4.4 Core functional module 44

4.4.1 Image upload component 45

4.4.2 Object detection with tensorflow.js 45

4.4.3 Traffic sign detection via Hugging face API 45

4.4.4 Confusion matrix 46

4.4.5 Classification report 48

4.4.6 Dashboard detection and rendering 50

4.5 Styling and Layout 51

4.5.1 Material-UI (MUI) 51

4.5.2 Styled-components 51

4.5.3 Raw CSS (app.css, index.css) 52

4.6 ESlint and code quality 52

4.7 Build and Deployment 52

4.8 User Interface 53

4.9 Future Enhancement 54

CHAPTER FIVE: CONCLUSION, CHALLENGES AND RECOMMENDATION

5.1 Summary 56

5.2 Conclusion 56

5.3 Challenges 57

5.4 Recommendation 58

REFERENCE 60

APPENDIX 62-64

**LIST OF FIGURES**

Fig. 2.1 Conceptual breakdown of adversarial attacks 15

Fig. 2.2 Overview of autonomous vehicle Inputs from camera and other sensors 21

Fig. 2.3 Types of adversarial attacks and countermeasures analysis systematic review 26

Fig. 3.1 System architecture of AI-based detection of adversarial image 29

Fig. 4.1 Hugging face access token user interface 44

Fig. 4.1 Image Upload component 44

Fig. 4.2. Object detection with tensorflow.js 45

Fig. 4.3 Traffic sign detection via hugging face 45

Fig. 4.4 Hugging face access token interface 46

Fig. 4.5 Confusion matrix 47

Fig. 4.6. Dashboard detector and rendering 51

Fig. 4.7. Deployment page 53

Fig. 4.8. User Interface 54

APPENDIX 62-64

**LIST OF TABLES**

Table 2.1 Taxonomy of Adversarial Attack Algorithms against Image-Detection Models 17

Table 2.2 Summary and comparison of various attack methods 23

Table 3.1 Summary of the system architecture 32

Table 3.2 Output and response handling 38

Table 4.1 Development environment and tools 42

Table 4.2 Traffic signs classification report 48

**LIST OF ABBREVIATIONS**

3D- Three dimensions

AI- Artificial Intelligence

Avs- Autonomous vehicles

CARLA- Car Learning to Act

CNN- Convolutional neutral networks

RNN- Recurrent Neutral Networks

GAN- Generative Adversarial Networks

ADS- Autonomous Driving Systems

IOT- Internet of Things

UAV- Unnamed Aerial Vehicles

LIDAR- Light Detection and Ranging

SAE- Society of Automotive Engineers

CAV- Connected and Autonomous Vehicles

PCS- Perception and Control systems

IDS- Intrusion Detection System

FGSM- Fast Gradient Sign Method

PGD- Projected Gradient Descent

C&W- Carlini & Wagner

XAI- Explainable Artificial Intelligence

V2X- Vehicle-to-Everything

ITFGSM- Iterative Targeted Fast Gradient Sign Method

Opt-uni- Optimization Universal Perturbation

AdvGAN- Adversarial Generative Adversarial Network

WB- White Box

BB- Black Box

FAB- Fast Adaptive Boundary Attack

HAA- Hierarchical Adversarial Attack

JSMA- Maximal Jacobian-based Saliency Map Attack

StAdv- Spatially Transformed Network

DNN- Deep Neutral Network

PRPEN- Pinpont Region probability estimation network

GNP- Gradient Norm penalty attack

ATTA- Adversarial Task-Transferable Attack

TAA- Targeted Attention Attack

CAM- Class Activation Mapping

ARP- Adversarial Retroreflective Patch Attack

ZOO- Zeroth-Order Optimization

TPA- Time-aware Perception Attack

GAN- Generative Adversarial Networks

IT-FGSM- Iterative Targeted Fast Gradient Sign Method

GPU- Graphics Processing Unit

ViTs- Vision Transformers

GTSRB- German Traffic Sign Recognition Benchmark

CLS- Classification token

API- Application programming interface

MUI- Material UI

HMR- Hot Module Replacement

CSS- Cascading style sheets

URL- Uniform Resource Locator

COCO-SSD- Common Object in Context- Single Shot Multibox detection

**CHAPTER ONE**

**INTRODUCTION**

**1.1 BACKGROUND OF STUDY**

In recent times, there has been significant growth in the advancement of autonomous driving technology and the integration of autonomous driving systems (ADS) into our traffic infrastructure. This progress can be attributed to the rapid development of artificial intelligence (AI) and Internet of Things (IoT) technologies. The implementation of AD technology has the potential to transform the transportation system by improving efficiency, reducing traffic congestion, minimizing human error, and enhancing safety on the roads. The importance of environmental image analysis in autonomous vehicle development and operation cannot be overstated. It is crucial for their perception and decision-making abilities, allowing for safe and efficient navigation. Autonomous vehicles, including cars, ships, underwater vehicles, unmanned aerial vehicles (UAVs), and robots, utilize a variety of sensors such as cameras, and LIDAR (Light detection and ranging), to capture detailed images and environmental data. These images are then processed using advanced algorithms and machine learning techniques to identify and track objects like pedestrians, cyclists, vehicles, traffic signs, and lane markings.  
Key players in the industry such as Tesla, Uber, General Motors, Rivian, and Microsoft-Waymo are making substantial contributions to the development of ADSs. These companies have already introduced services and created a platform for the commercialization of these systems. While most of the advancements are currently at level 3 AD technology, where the vehicle can handle most driving tasks but still requires human intervention, there are challenges related to safety and reliability due to the evolving nature of ADSs operating in diverse and dynamic environments. These challenges are hindering the progress towards achieving level-five self-driving technology.  
The rise of IoT technology has opened up opportunities for the development of intelligent real-world applications, leading to the creation of a smart driving environment. The increasing demand for advanced data processing capabilities in ADSs aligns with the growing number of IoT devices such as smart traffic lights, signs, and boards enabled by IoT technology. These smart devices have enhanced environmental awareness among ADSs, enabling them to interact with their surroundings and make informed driving decisions. For instance, in situations like fog or heavy rainfall where the ADS's camera may not provide clear visibility, vehicles can communicate with other IoT devices in the area to gather additional data and improve their navigation. Ensuring the security and resilience of the interconnected network of devices within the driving environment is crucial to safeguarding the integrity and reliability of ADSs. Addressing vulnerabilities to potential attacks is a top priority in this regard.  
  
The Society of Automotive Engineers (SAE) categorizes autonomous vehicles into different levels of automation, from level 0 (no automation) to level 5 (full automation). As the level of automation increases, so does the system's independence and the reduction of human intervention, largely due to advancements in image analysis technology. At higher levels of automation, such as level 4 (High Automation) and level 5 (Full Automation), vehicles operate without human intervention in most scenarios, relying on robust image analysis systems that combine visual data with inputs from other sensors like LiDAR and radar. (Yao Deng, Jin, and Qing-Long Han. 2021. Deep learning-based autonomous driving systems: A survey of attacks and defenses. IEEE Transactions on Industrial Informatics 17, 12 (2021))

Moreover, environmental image analysis plays a critical role in the advancement of autonomous vehicles, enabling them to operate independently and safely in various conditions. The evolution of image analysis technology has paved the way for higher levels of automation, where vehicles can handle all driving aspects without human oversight, showcasing the importance of sophisticated image analysis capabilities in the realm of autonomous vehicles.

**1.2 PROBLEM STATEMENT**

Undoubtedly, the issue of adversarial attacks against connected and autonomous vehicles (CAVs) or Autonomous driving systems (ADS) remains a significant area of focus in both research and practical applications. Despite the progress made in understanding these attacks, existing research has focused predominantly on manipulating perception models in CAVs, driven by the significant success of adversarial attacks in the image domain either physically or as a result of system malfunction. Perception and control systems (PCSs) enable vehicles to understand the environment by analyzing data from sensors like cameras and LiDAR, facilitating autonomous navigation. However, they do not explore other potential models within Autonomous Driving system where adversarial attacks could be equally impactful, such as those related to cybersecurity tasks. Therefore, our work aimed to bridge this gap and expand the threat landscape of adversarial manipulation in autonomous vehicles. Specifically, we investigated the susceptibility of IDS models deployed within CAVs to such attacks, thus providing a more comprehensive perspective on the adversarial attacks faced by the vehicle.

**1.3 SIGNIFICANCE OF THE STUDY**

In order to provide intelligent features and smooth integration with the vast diversity of digital services that users utilize, autonomous driving systems (ADS) are outfitted with an increasing number of digital systems and internet connectivity. The increasing digitalization of automobiles creates a wider attack surface that malevolent actors motivated by financial gain (e.g., ransomware) or the desire to inflict bodily harm might take advantage of. Automated vehicles need therefore to be properly secured to mitigate the safety risks caused by malicious actions, due to their integration in a high-risk environment. This tendency is accelerated by the growing use of AI technologies in automobiles, which add another level of complexity to traditional software components. This more complex and larger digital ecosystem in vehicles raises significant cybersecurity challenges, particularly considering

the potential impact a Adversarial attacks could have if the core functionalities of the vehicle are compromised

The inclusion of more advanced AI components in higher levels of automation further complicates this picture by introducing a whole new range of potential vulnerabilities, the malevolent exploitation of which could cause intended offence and harm. Recent years have seen an increasing amount of research and practical examples highlighting new attacks against machine learning systems.

**1.4 AIM AND OBJECTIVE**

**1.4.1 AIM**

The creation of an AI-driven framework for the identification and defense against adversarial image attacks directed at autonomous systems (connected and autonomous vehicles (CAVs)) is the main goal of this project. By using machine learning techniques to detect, categorize, and eliminate hostile perturbations, the study aims to improve the security and resilience of computer vision models used in connected and autonomous vehicles and other self-driving automobiles. This research will explore the effectiveness of deep learning, feature extraction, and ensemble learning strategies to improve detection accuracy. Additionally, it aims to provide a scalable and adaptive solution capable of real-time implementation in real-world scenarios, ensuring the safe operation of autonomous systems under potential adversarial threats.

**1.4.2 OBJECTIVE**

To achieve the aim, this project has specific objectives:

* Comprehending Adversarial Attacks on Self-Sustained Systems by examining the characteristics and effects of adversarial image attacks on autonomous systems' computer vision models.
* Creation of a Detection Framework Based on AI by Designing and implementing deep learning-based detection models to identify adversarial perturbations in image inputs as well as comparison of traditional detection methods with AI-driven approaches to evaluate efficiency and effectiveness.
* Adversarial Image Feature Extraction and Categorization by differentiating hostile images from benign ones, use feature extraction methods like frequency domain analysis and convolutional neural networks (CNNs) with the classification of algorithms based on various adversarial attack types.
* Enhancing Model Robustness Against Adversarial Perturbations by Investigating adversarial training and data augmentation techniques to improve model resilience. These include Exploring defensive mechanisms such as input preprocessing, gradient masking, and ensemble learning strategies.
* Validation of Experiments and Assessment of Performance by testing AI-based detection framework on simulated and real-world autonomous systems, such as self-driving car prototypes or UAVs with the aim of accessing the feasibility and integration of the proposed solution in commercial autonomous platforms.

**1.5 METHODOLOGY**

For an image classification model, an adversarial attack is considered successful if an adversarial image is classified as a different class compared with the original image. However, autonomous driving models are regression models that predict continuous values. Therefore, adversarial attacks on driving models are defined with respect to an acceptable error range, known as adversarial threshold. Hence, an adversarial attack on a driving model is considered successful if the deviation between the original prediction and the prediction of an adversarial example is above the adversarial threshold.

Current adversarial attacks on classification model could be categorized into three classes based on the perturbation generation method. Fast Gradient Sign based method directly generates adversarial examples by adding the sign of the loss gradient with respect to each pixel on original images. Optimization-based method formulates the adversarial example construction as an optimization problem. Generative model based method proposes to generate adversarial examples by harnessing the power of generative models.

This study employs a systematic approach that includes data gathering, model building, the creation of adversarial attacks, the construction of detection frameworks, and experimental assessment. Key phases of the approach are as follows:

1. **Data Collection and Preprocessing**

* Utilize benchmark datasets such as ImageNet, and real-world autonomous driving datasets (e.g., KITTI, Waymo Open Dataset, Huggingface API).
* Apply image preprocessing techniques, including normalization, resizing, and noise reduction, to enhance data quality.
* Augment the dataset with variations to improve model generalization and robustness.

**2.** **Adversarial Attack Generation**

* Implement adversarial attack techniques, including FGSM, PGD, and C&W, to perturb input images.
* Simulate real-world attack scenarios to test the vulnerability of autonomous system perception models.
* Create a diverse adversarial dataset for training and testing purposes.

1. **Development of AI-Based Detection Framework**

* Design deep learning-based models (CNNs, ResNet, and Vision Transformers) for adversarial detection.
* Explore hybrid detection methods that integrate supervised and unsupervised learning.
* Apply feature extraction techniques, such as wavelet transformations and frequency domain analysis, to detect adversarial perturbations.

1. **Model Training and Optimization**

* Train the detection models using adversarial and clean image datasets.
* Implement optimization techniques such as learning rate scheduling, dropout, and batch normalization.
* Utilize adversarial training to improve model robustness against attacks.

1. **Performance Evaluation**

* Measure model performance using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC.
* Conduct ablation studies to evaluate the effectiveness of different detection techniques.
* Compare the proposed approach with existing adversarial detection methods.

1. **Real-World Testing and Deployment**

* Implement the detection framework on real-world autonomous systems, such as UAVs and self-driving car simulations.
* Evaluate real-time detection efficiency under different environmental conditions.
* Assess computational constraints and optimize the model for edge and cloud-based deployment.

1. **Security Analysis and Future Enhancements**

* Analyze potential weaknesses of the proposed framework and suggest improvements.
* Investigate explainable AI (XAI) methods to enhance interpretability.

This methodology ensures a comprehensive approach to detecting adversarial image attacks, improving the security of AI-based autonomous systems, and facilitating real-world applicability.

**1.6 SCOPE OF STUDY AND LIMITATIONS**

**1.6.1 SCOPE OF STUDY**

This study focuses on the detection and mitigation of adversarial image attacks on autonomous systems. It covers various attack strategies, AI-based detection methodologies, and model training approaches to enhance robustness. The research is limited to image-based adversarial attacks and does not extend to other modalities such as LiDAR or radar-based attacks. The scope also includes real-world validation using autonomous vehicle and UAV datasets. These explorations will cover recent breakthroughs in these domains, such as innovations in transfer learning, multitask learning, and unsupervised learning methodologies. By analyzing the latest research findings and technological developments, the article will highlight how these AI solutions enhance the environmental perception capabilities of autonomous vehicles, facilitating tasks like semantic segmentation, 3d object detection and trajectory prediction.

Ultimately, by synthesizing information on both datasets and AI solutions, the article aims to provide a valuable resource for researchers, developers, and industry stakeholders, offering insights into the current landscape and future directions of image analysis in autonomous vehicle technology.

Also, the purpose of this article is to conduct a comparative analysis of the effectiveness and application of various AI techniques in real-world scenarios, with a particular focus on three critical tasks: semantic image segmentation, 3D object detection in video, and vehicle control in virtual environments. This involves detailed examination of how different AI methodologies perform across these key functions, which are essential for the robust operation of autonomous vehicles.

**1.6.2 LIMITATIONS**

Despite these advances, several limitations remain impeding the full realization of autonomous driving:

**1.** **Annotated datasets**

While expansive datasets like Cityscapes and nuScenes render robust model training, the necessity for substantial human annotation remains a bottleneck. Data labels must be precise, consistent, and extensive to ensure comprehensive model training, yet the manual effort required is both time-consuming and expensive.

**2.** **Generalization**

Models trained on specific datasets often struggle to generalize effectively across different environments and conditions. Differences in weather, road conditions, and regions can impact the performance of perception systems, necessitating continual data collection and model retraining.

**3.** **Real-time processing**

Achieving high processing speeds without compromising detection and classification accuracy is a challenging trade-off. This becomes critical in real-time applications where latency must be minimized to ensure safety and reliability.

**4.** **Biases in AI models**

The models may inadvertently incorporate biases from training datasets, resulting in systemic prejudices that can affect decision-making. Therefore, identifying and mitigating these biases remains a significant challenge.

**1.7 FUTURE RESEARCH TRENDS**

**1.** **Advanced data collection and annotation**

The development of semi-automated and fully automated annotation tools could alleviate the cumbersome data labeling process. Techniques like weak supervision and active learning can substantially reduce human effort while enhancing the quality and diversity of training datasets.

**2.** **Domain adaptation and generalization**

Future research could focus on developing models that generalize better across diverse environments. Domain adaptation techniques and unsupervised learning approaches could improve the robustness of AI models, enabling them to perform consistently in varying real-world scenarios.

**3.** **Real-time processing enhancements**

Optimizing network architectures to balance accuracy with processing speed will be crucial. Leveraging hardware advancements alongside software optimizations can lead to significant improvements in the real-time application of these models.

**4.** **Mitigating AI biases**

Developing techniques to identify, quantify, and mitigate biases within datasets and AI models will be an essential area of future research. Enhancing the transparency and interpretability of AI models can lead to more equitable and trustworthy machine learning systems.

**5.** **Integration of novel sensor data**

Expanding the use of multimodal sensors, including radar and advanced LiDAR systems, coupled with innovative data fusion algorithms, can enhance the perception capabilities. Collaborative perception and V2X (Vehicle-to-Everything) communication technologies represent promising avenues for research.

**6.** **Simulation and virtual environments**

Refining simulation tools like CARLA and integrating them with real-world data can create more effective and versatile training systems for autonomous vehicles. These simulators will be critical for validating algorithms under varied and controlled conditions.

**7.** **Human-machine interaction**

Improving the interface between human operators and autonomous systems can enhance safety and trust. Research focusing on intuitive control mechanisms and fail-safe protocols is essential to ensure effective human intervention when necessary.

In summary, while substantial progress has been made in leveraging AI for image analysis in autonomous vehicles, ongoing research and innovation are required to overcome existing limitations and pave the way for more advanced, reliable, and safe autonomous driving systems.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 INTRODUCTION**

Many adversarial attacks have been proposed and demonstrated effectively on image classification models. To defend against adversarial attacks, several techniques have been proposed to harden neural networks. However, previous research mainly focuses on image classification models. It is unclear to what extent these adversarial attacks and defenses are effective on regression models (e.g., autonomous driving models as well as connected and autonomous vehicles). This uncertainty exposes potential security risks and raises research opportunities. If adversarial attacks could be successfully applied to autonomous driving systems, attackers could easily cause traffic accidents and jeopardize personal safety. If existing defense methods cannot be adapted to defend against attacks on regression models, it is imperative to identify a novel defense mechanism suitable for autonomous driving.

This paper basically presents a comprehensive analysis on various adversarial attack methods as well as defense methods on autonomous driving models. By conducting systematic experiments on three driving models, we find that except ITFGSM (Iterative Targeted Fast Gradient Sign Method) has a 36% attack success rate, all other attacks including Opt-uni (Optimization Universal Perturbation) and AdvGAN (Adversarial Generative Adversarial Network) could effectively generate adversarial examples with an average of 98% success rate in the white-box setting. Therefore, like classification models, CNN-based (Convolutional Neutral Network) regression models are also highly vulnerable to adversarial attacks. On the other hand, the attack success rate of all attack methods is significantly lower in the black-box setting (4% only on average). This implies that, if neural network architecture and hyperparameters are not known, a driving model is much less vulnerable to adversarial attacks.

Many studies have been conducted on defensive strategies against adversarial attacks, and current research has found that deep learning models are not completely secure because an attacker can input adversarial examples, which are specifically designed perturbations to cause these deep learning models to predict erroneously. These perturbations are undetectable to humans, but they are strong enough to fool the model reliably. An attacker can execute attacks on a machine learning model in two main stages in a machine learning pipeline. The first category is the machine learning model-training phase attacks. This can be further divided into three main types. The first type of attack is known as the data injection attack where the attacker injects adversarial samples into the training dataset to change the distribution by poisoning but without any knowledge about the training dataset. The second type is data modification attacks where the adversary is modifying or contaminating the training dataset with knowledge of it. In these two types of attacks, it assumes that the adversary has no knowledge about the target model. The latter type of attack is known as the logic corruption attack where the attacker tries to modify the target model. It is assumed here that the attacker is fully aware of the model. The second type of attack is testing phase attacks. During this stage, the attacker primarily causes misclassifications in the model output by generating adversarial perturbations. From the above two main categories (i.e., training phase and testing phase attacks), testing phase attacks get higher attention because there are many studies conducted for attacks and defense methods at this stage. Attacks during the testing phase are further classified into three categories: black-box, white-box, and grey-box attacks. In black-box attacks, it is assumed that the attacker does not know the model, in white-box attacks, it is assumed that the attacker has full knowledge of the model, including architecture and defense methods, and in the grey-box attacks, the attacker has some knowledge about the model such as the structure of the model and training data but no knowledge about the weights of the model. Considering the adversary’s knowledge about the network, the white-box attacks could be stated as the strongest and the black box attacks are the weakest ones.

A diagram of a machine learning

AI-generated content may be incorrect.

Fig. 2.1. Conceptual breakdown of the adversarial attacks.

**2.2 AUTONOMOUS DRIVING MODELS**

Details from an autonomous driving model shows the input data from sensors (e.g., LiDARs and cameras), a deep neural network predicts the control of the vehicle such as the steering angle and speed. CNN is the mainstream neural network architecture for autonomous driving, since it has excellent performance, requiring less neurons and consuming less resources. In autonomous vehicles, such driving model is usually included inside a perception domain controller, which can be updated remotely through the vehicle’s gateway. Some companies have published their research on autonomous driving. For example, Comma.ai, an American technology company that specializes in developing autonomous driving software presents a CNN based driving model to predict the steering angle based on driving video. Nvidia also a technology company specializing in graphics processing units (GPUs) as well as autonomous driving technologies builds a CNN model called DAVE-2. They demonstrate that DAVE-2 can automatically drive a vehicle without human intervention 90% of the time in a simulation test while performing autonomous steering 98% of time in an on-road test. (Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. CoRR, abs/1604.07316, 2016)

**2.3 ADVERSARIAL ATTACKS**

By adding small perturbations to original images, adversarial attacks can deceive a target model to produce completely wrong predictions. Currently, adversarial attacks are mainly researched on image classification tasks. Given a target model and an original image with its class, an adversarial attack constructs an imperceptible adversarial perturbation to form an adversarial example and make the target model classify the model which is different from the original.

Depending on the information required to perform the attack, existing adversarial attacks can be categorized into white-box attacks and black-box attacks. White-box attacks require all details of a target model to perform the attack, including the training data, the neural network architecture, parameters, and hyper-parameters, as well as the privilege to gather the gradients and prediction results of a model. By contrast, black-box attacks only require to query the model with arbitrary input data and get the prediction result. Based on the inputs and outputs from the target model, to perform a black-box attack, attackers can build a substitute model and achieve white-box attacks on their own model. The adversarial examples on the substitute model could then be used to attack the target black-box model, which is called the transferability of adversarial examples.

White-box (WB) Attacks. White-box attacks are scenarios where adversaries have complete knowledge about and access to the target model and its insights, such as training samples, design, parameters, gradient information, and defense scheme. Such attacks usually introduce optimal vulnerabilities in the model during the training phase, giving it a high probability of being deceived at the inference phase. However, model designers often use white-box attacks for robustness assessments and verifications of DL models. Real-world attackers rarely succeed in misleading target models using white-box attacks due to the need for complete access to and knowledge of the target model, which is not often practical in real-world settings. (S. Asha and P. Vinod. 2022. Evaluation of adversarial machine learning tools for securing AI systems. Cluster Computing 25, 1 (2022), 503–522)

Table 2.1. Taxonomy of Adversarial Attack Algorithms against Image-Detection Models

|  |  |  |  |
| --- | --- | --- | --- |
| ATTACK ALGORITHMS | ATTACK SCHEME | ADVERSARY’S INTENTION | ADVERSARY’S AWARENESS |
| Fast adaptive boundary attack (FAB) | Gradient | Untargeted | WB |
| Houdini | Gradient | Tarfeted/untargeted | BB |
| Hierarchical adversarial attack (HAA) | Gradient | Untargeted | WB |
| Maximal Jacobian-based Saliency Map (JSMA) Attack | Gradient | Targeted | WB |
| Cui et al | Gradient | Targeted/untargeted | WB |
| Translation Invariant-based Attack | Geometry | Targeted/untargeted | WB |
| Spatially Transformed Network (stAdv) | Geometry | Targeted | WB |
| Adaptive local attack | Geometry | Targeted/untargeted | WB |
| Mani fool | Geometry | Targeted/untargeted | WB |
| Carlini-wagner (C&W) attack | Optimization/Gradient | Targeted/untargeted | WB/BB |
| GenGrad attack | Optimization | Targeted | BB |
| Scene-specific attack | Geometry | Targeted/untargeted | WB |
| Scene Agnostics adversarial patch attack | Optimization/Transfer | Targeted | WB |
| L 1- Oriented Elastic-net attacks to DNNs | Transfer | Targeted | WB/BB |
| Feature-aware Transferable attack | Transfer | Targeted | WB/BB |
| Robust physical Pertubation (RP2) | Transfer/Gradient | Targeted/untargeted | WB/BB |
| Neutral light illuminations- based attack | Transfer/Geometric | Targeted/untargeted | WB/BB |
| Min-Max optimization based attack | Optimization/gradient | Targeted/untargeted | BB |
| Pinpont Region probability estimation network (PRPEN) | Transfer | Targeted/untargeted | BB |
| Gradient Norm penalty (GNP) attack | Gradient/Transfer | Untargeted | WB/BB |
| Adversarial Task-Transferable Attack (ATTA) | Transfer | Untargeted | BB |
| Dynamic Adversarial Attacks | Transfer/Score | Targeted/Untargeted | BB |
| Targeted Attention Attack (TAA) | Transfer/Decision | Targeted | BB |
| PhysGAN | Transfer/Score | Targeted/Untargeted | BB |
| DeepBillboard | Transfer | Targeted | BB |
| Adversarial Patch Attack on ADSs | Transfer/Score | Targeted | BB |
| Class Activation Mapping (CAM) | Decision | Targeted | BB |
| Physical One-Pixel Attack | Transfer/Gradient | Targeted/Untargeted | BB/WB |
| Adaptive Square Attack | Transfer/Score | Targeted | BB |
| Adversarial Retroreflective Patch (ARP) Attack | Score/Geometric | Targeted/Untargeted | BB/WB |
| Zeroth-Order Optimization (ZOO) | Transfer/Score | Targeted/Untargeted | BB |
| Liu et al | Gradient/Transfer | Targeted/Untargeted | BB/WB |
| Time-aware Perception Attack (TPA) | Score | Targeted/Untargeted | BB |
| OptiCloak | Score/Gradient | Untargeted | BB/WB |
| Multi-source Adversarial Attack Models | Gradient/Score | Untargeted | BB/WB |
| Out-of-distribution Attack | Decision | Targeted/Untargeted | WB/BB |
| GhostStripe Attack | Decision/Geometric | Targeted/Untargeted | BB,WB |
| D-BADGE Attack | Decision | Targeted/Untargeted | BB |
| ShapeShifter | Decision | Targeted/Untargeted | WB |
| Robust Disappearance Attacks | Decision | Targeted | WB/BB |
| GenAttack | Decision | Targeted | BB |

**2.4 DEFENDING AGAINST ADVERSARIAL ATTACKS**

There are many attacks introduced in each of the above discussed stages. Moreover, when defending against these adversarial attacks, the researchers introduce several defense strategies. Among such strategies, the adversarial training method where the model is re-trained by augmenting adversarial examples to the training dataset with their correct labels is widely used. Here, adversarial perturbations are generated by selecting one or many attacks. Another popular defense mechanism is the defensive distillation method. The main objective of the defensive distillation method is to make the learning process smooth and remove the volume of gradients around the inputs. Apart from these, Generative Adversarial Networks (GAN) based approaches (Defense GAN) and denoiser-based defense approaches have been introduced. Nevertheless, adversarial training is the most promising adversarial defense approach and several seminal improvements have been carried out and introduced. These adversarial defense methods into three main strategies. The first one is modifying the databased defense method, which refers to modifying both training and testing phase data and improving the robustness of the models. Defensive techniques such as adversarial training, gradient hiding and input transformation belong to this method. The second strategy is modifying the model-based defenses. This strategy includes defense mechanisms like defensive distillation and regularization. The last one is auxiliary tool-based defense models, which use additional tools such as GAN networks in the defense model. This is the summary of the adversarial attacks and the defense models commonly used in present. ( Haibo Zhang, Zhihua Yao, and Kouichi Sakurai. 2024. Versatile defense against adversarial attacks on image recognition. arXiv:2403.08170 2024)

The security threat of the adversarial attack has received more attention among the research community with the arrival of AVs, which detect and classify objects, control speed and plan paths via the use of deep neural networks. This is understandable given the growth of computer vision technologies in AVs based on machine learning, and the future of autonomous and unmanned vehicles is likely to be based on machine learning. Thus, these adversarial attacks would be a significant threat. At present, there are several research works devoted to introducing novel adversarial attacks and defense models, pipelines in AVs. However, there is still a technical barrier to making fully robust defense models against these adversarial attacks.

A close-up of a car

AI-generated content may be incorrect.

Fig. 2.2. Overview of autonomous vehicle Inputs from camera and other sensors.

**2.5 ADVERSARIAL ATTACK METHODS ON AUTONOMOUS DRIVING SYSTEMS**

An adversarial attack is considered successful if an adversarial image is classified as a different class compared with the original image. However, autonomous driving models are regression models that predict continuous values. Therefore, adversarial attacks on driving models are defined with respect to an acceptable error range, known as adversarial threshold. Hence, an adversarial attack on a driving model is considered successful if the deviation between the original prediction and the prediction of an adversarial example is above the adversarial threshold.

Current adversarial attacks on classification models could be categorized into three classes based on the perturbation generation method. Fast Gradient Sign based method directly generates adversarial examples by adding the sign of the loss gradient with respect to each pixel on original images. Optimization-based method formulates the adversarial example construction as an optimization problem. Generative model-based method proposes to generate adversarial examples by harnessing the power of generative models such as Generative Adversarial Network (GAN) and autoencoder networks. In addition, there is a special attack named Universal attack, which generates a single adversarial example to fail all samples in the dataset.

However, there are major re-implementation of adversarial attacks to form a comprehensive set of adversarial attacks on regression models. We first choose two classic adversarial attacks: Iterative Targeted Fast Gradient Sign Method (IT-FGSM), a variant of the classic method Fast Gradient Sign Method (FGSM), and an optimization-based approach as it is the first approach to generate adversarial examples. We then choose a state-of-the-art generative model-based attack called AdvGAN. Furthermore, we implement two universal attack methods to increase the diversity of attacks in our experiments. We do not choose attack methods such as C&W attack and DeepFool, since these attacks rely on the attributes of classification models (e.g. decision boundary and the Softmax function). Thus, they cannot be adapted to regression models. We elaborate on the five selected attack methods below.

1. ITERATIVE TARGETED FAST GRADIENT SIGN METHOD (ITFGSM): This is a white-box attack that requires model knowledge. It is a variant of Fast Gradient Sign Method (FGSM) that simply adds the sign of the loss gradient with respect to each pixel on original images. ITFGSM applies the targeted FGSM multiple times to get a more powerful adversarial example. It can fool object detection models making a stop sign appear as a speed limit sign

2. OPTIMIZATION-BASED APPROACH (OPT): this can either be a white-box or a black-box attack depending on the implemenataion. This approach calculates an adversarial perturbation using mathematical optimization technique to find the smallest possible perturbation that deceives the model. It is mostlu used to attack image classifiers and object detectors in self-driven systems.

3. AdvGAN: AdvGAN can either be a white-box or black-box attack using GANs. It generates an adversarial example from an original image by integrating another objective into the objective function. Once trained, the GAN can generate perturbation in real-time, making it faster than iterative gradient-based attacks. AdvGAN is unique, unlike other forms of attacks,it doesn’t need to compute gradient each time, making it efficient.

4. UNIVERASAL ADVERSARIAL PERUTRBATION (OPT UNI): Implementation of attack is based on the optimization based approach. Opt uni creates a single, optimized perturbation that can fool many images. The main aim is to corrupt entire datasets used in training or interference. The perturbation may seem small but they are broadly effective.

5. AdvGAN UNIVERSAL ADVERSARIAL PERTURBATION (AdvGAN uni): This generates Universal perturbation, i.e. it can attack multiple inputs rather than crafting unique perturbation for each image.The GAN produces one perturbation that can generalize across different image and evn different models which may be highly dangerous for real-word adversarial attacks and can disrupt an entire system. It is more efficient than opt-uni and automatically generates a strategy rather than relying on optimization. In this study, we implement this approach based on the AdvGAN architecture.

Table 2.2. Summary and comparison of various attack methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATTACK METHOD** | **TYPE** | **SPEED** | **GENERALIZATION** | **REAL-WORLD APPLICATION** |
| ITFGSM | Targeted Iterative Attack | Medium | Low(per image) | Moderate( image specific miscalassification) |
| Optimization-based | White-box optimization | Slow | Medium | High (Precise but computationally expensive) |
| AdvGAN | GAN-Based attack | Fast | Low( per-image) | Very High (efficient and stealthy) |
| Opt-Uni | Universal Perturbation | Mediun | High( one perturbation for many images) | High (Broad attacks possible) |
| AdvGAN uni | Universal GAN-Based attack | Very fast | Very high | Extremely high (Adaptive, efficient and hard to detect |

**2.6 ADVERSARIAL ROBUSTNESS AND DEFENSE STRATEGIES**

Implementing a general defense framework that addresses the vulnerability of both man-made adversarial attacks and physical world adversarial corruptions at the same time would be a promising open research problem, because it would save the cost and the complexity of the system which performs the main task integrated with several auxiliary tasks to improve domain knowledge of the prediction via different factors. This could be identified as recent research on general adversarial robustness. However, since this method uses multiple auxiliary networks during the inference phase, resource consumption of this approach has to be evaluated. This is due to the fact when it comes to the autonomous driving systems improving the general resilience without using any supporting tools in the inference is essential due to the resource constraints.

Adaptation and re-implementation of four defense methods for autonomous driving. First, we choose two classic proactive defense methods, Then we develop a reactive defense method based on the insight that real-time adversarial example generation may lead to a resource usage spike in autonomous vehicles. (Nicolas Papernot, Patrick McDaniel, Somesh Jha, and Anan thram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In Proc. of S&P,. IEEE, 2016)

**2.6.1 PROACTIVE:**

1) Adversarial Training: By retraining the original model with adversarial examples, the new model learns features of adversarial examples and thus has better generalization and robustness. In this study, we add adversarial examples generated by proposed attacks to train a new model.

2) Defensive Distillation: Defensive distillation uses class probabilities predicted by the original model as soft labels to train a new model. The adaptation of the original defensive distillation approach to handle regression models based on research shows that the output of a neural network hidden layer contains highly encoded information that can be leveraged for model distillation. Similarly, the output of fully connected layers can be used to perform defensive distillation on CNN based driving models.

**2.6.2 REACTIVE:**

1) Anomaly Detection: Autonomous vehicles are usually equipped with a runtime monitoring system to check the vehicle state. First, we monitor model prediction latency caused by adversarial attacks. Second, since autonomous vehicles are resource constrained, we monitor any spikes in GPU memory usage and GPU utilization rate via a System Management Interface to detect additional computation caused by adversarial attacks. We evaluate the effectiveness of this anomaly detection approach by comparing the prediction time per image and GPU usage with and without adversarial attacks and further investigate its effectiveness on different kinds of attack methods.

2) Feature squeezing: There are two proposed feature squeezing methods for adversarial defense. The first method squeezes the original 24-bit color down to 1 bit to 8 bit color. By doing so, adversarial noise becomes more perceptible as the bit depth decreases. The second method adopts median spatial smoothing, which moves a filter across an original image and modifies the center pixel value to the median of the pixel values in the filter. If the difference between the prediction results of the original image and the prediction result of a squeezed image by either of the two methods exceeds a predefined threshold, then the given input is likely to be an adversarial example.

A diagram of a structure of attack

AI-generated content may be incorrect.

Fig. 2.3. Types of adversarial attacks and countermeasures analysis systematic review.

**2.7 INCREASING THE SPEED AND EFFICIENCY OF THE DEFENSE MODELS**

The ecosystem of AVs is connected to cloud computing technology. As discussed in defense methods introduced in AVs, some defense methods get considerable time to perform against adversarial attacks. However, real-time combating of adversarial perturbations is essential in a safety-critical domain like AVs. To improve the efficiency of executing adversarial robust machine learning models, this paper has proposed using innovative technologies like 5G. Moreover, since AVs have complicated intelligent components, built lightweight adversarial defense methods and improving the resilience of the existing models naturally without changing the network architecture or without using any supporting tool in the inference would be some promising research areas.

**CHAPTER THREE**

**METHODOLOGY**

**3.1 INTRODUCTION**

This chapter indicates the methodology for developing an AI-based detection system for adversarial image attacks on self-driven autonomous system. Its detailed proposed approach includes dataset acquisition, prepossessing, model selection, training procedures, evaluation matrices and implementation strategies. Given the limitations of traditional convolutional neutral network (CNN) in detecting adversarial perturbations. The structured methodology ensures a systematic approach to designing a robust adversarial image detection system, enhancing the security and reliability of the autonomous systems. This project helps develop a robust AI-based system by leveraging Vision Transformers (ViTs) for enhanced robustness and also creating a dataset for adversarial perturbated images to integrate the detection system into the autonomous vehicle framework.

**3.2 SYSTEM ARCHITECTURE**

The AI-based adversarial image detection system's architecture aims to improve autonomous cars' visual perception dependability. As deep learning models are used more often in self-driving systems, adversarial assaults are a significant risk because they can subtly change input photos to produce inaccurate predictions and possibly safety-critical errors. By combining cutting-edge computer vision algorithms with a strong detection pipeline, our technology is specifically designed to detect such manipulations and guarantee that autonomous vehicles base their choices on accurate visual data.

For scalability and efficiency, the suggested system uses a modular design, breaking up image acquisition, detection, and reaction into separate parts. In contrast to conventional convolutional neural networks, it makes use of a refined Vision Transformer (ViT) model because of its superior performance in managing spatial dependencies and adversarial robustness. From preprocessing raw image data to identifying if a picture is clean or adversarial and finally carrying out the proper system-level reactions, each module is essential. In addition to real-time attack detection, this design incorporates learning capabilities to adjust to changing adversarial tactics, strengthening the system's defenses against potential threats.

The proposed AI-based detection system for adversarial image attacks on autonomous systems consists of multiple interconnected components. These components work together to detect adversarial perturbations in images captured by an autonomous vehicle’s perception system. The architecture is divided into three main modules:

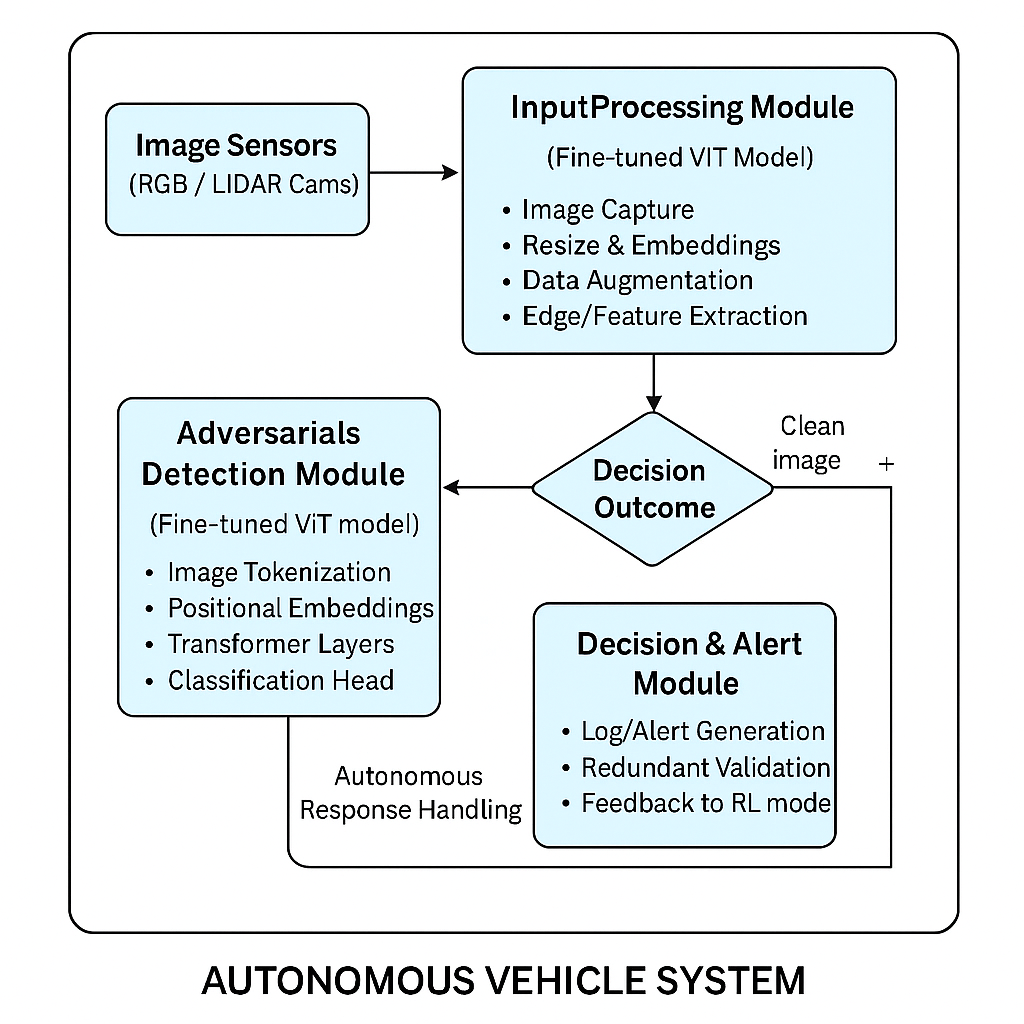


Fig.3.1 System architecture of AI-based detection of adversarial image

**3.2.1. INPUT IMAGE PROCESSING MODULE:** Captures and preprocess images before feeding them to the adversarial detection model. The Input Image Processing Module is responsible for receiving and preparing images for analysis. Key processes include:

* Image Capture: Images are collected from sensors such as RGB cameras or LiDAR-equipped cameras in real-time.
* Re-sizing & Normalization: Images are re-sized to a standard format (e.g., 224×224 pixels) and normalized to improve model stability.
* Data Augmentation: Random transformations such as rotation, contrast adjustment, and Gaussian noise are applied to enhance generalization.
* Edge and Feature Extraction: Edge detection (e.g., Sobel filters) and key feature extraction techniques are applied to improve robustness against minor adversarial manipulations.

Once preprocessing is complete, the images are passed to the detection module.

**3.2.2. ADVERSARIAL PERTUBATION DETECTION MODULE:** Uses a fine-tuned Vision Transformer (ViT) model to identify adversarial manipulated images. This is the core component of the system, utilizing a fine-tuned Vision Transformer (ViT) to detect adversarial perturbed images. It operates in the following steps:

* Image Tokenization: Unlike CNNs, which process entire images using convolutional layers, ViTs split images into fixed-size patches and process them as sequences. Each patch is flattened into a vector and fed into the transformer model.
* Feature Encoding & Classification: The model assigns positional embeddings3 to each patch to maintain spatial information. The patches are passed through multiple transformer layers, which capture long-range dependencies and relationships between different parts of the image.
* The final transformer output is processed by a classification head, which determines whether the image is clean or adversarially perturbed.

**Detection Outcomes**

The ViT model outputs a classification probability: If \( P\_{adv} > T \) (where \( P\_{adv} \) is the probability of adversarial perturbation and \( T \) is the detection threshold), the image is flagged as adversarial. If not, the image is considered clean and passed on for further processing.

**3.2.3. DECISION AND ALERT MODULE** – Determines appropriate responses to detected adversarial attacks and integrates corrective actions. If an adversarial attack is detected, the system triggers an appropriate response:

* Log & Alert Generation: The detected attack is logged with metadata (e.g., timestamp, camera location). An alert is sent to the autonomous system’s control unit, warning about the possibility of manipulated perception data.
* System-level response: For an image rejection, the system discards the manipulated image and requests a new frame. While, redundant processing analyzes image using additional detection techniques to confirm the attack. For the reinforcement learning feedback, the model updates itself based on new attack patterns improving detection accuracy.
* Autonomous Vehicle Reaction: If a speed limit sign is misclassified due to an attack, the vehicle avoids immediate changes in speed and if a stop sign is missing due to perturbation, the vehicle applies additional verification methods (e.g., radar or prior map data) before proceeding

**3.3 SUMMARY OF THE SYSTEM ARCHITECTURE**

Table 3.1 Summary of the system architecture

|  |  |
| --- | --- |
| **MODULE** | **FUNCTION** |
| Input image processing | Capture, preprocess and normalize image from the vehicle’s sensor |
| Adversarial Perturbation Detection | Uses a fine-tuned Vit model to classify image as clean or adversarial |
| Decision and alert system | Determines appropriate responses, logs events and alert the control unit |
| System integration | Ensures compatibility with autonomous vehicle frameworks and real-time processing |

**3.4 DATASET ACQUISITION AND PREPROCESSING**

The performance of an AI-based adversarial image detection system depends heavily on the quality and diversity of the dataset used for training and evaluation. This outlines the process of selecting, augmenting and generating adversarial examples to ensure a comprehensive dataset for training the Vision Transformer (ViT) model

**3.4.1 DATA SELECTION**

Originally, there are no publicly available datasets specifically designed for adversarial image attacks on autonomous perception systems, Since a dedicated dataset for adversarial attacks on traffic sign images is not readily available, this study shows different datasets which could be used but the main one adopted was the Hugging face’s Dima806/traffic\_sign\_detection.

**3.4.1.1 HUGGING FACE**

Hugging face is an open source platform best known for its work in machine learning (ML) and natural language processing (NPL) which provides tools, models, datasets, spaces, docs and libraries that make it easier for developers and researchers to build, train and deploy AI models. Hugging face datasets library offers a wide variety of curated datasets for training and evaluating machine learning models.

**3.4.1.2 HUGGING FACE MODEL *DIMA806/TRAFFIC\_SIGN\_ DETECTION***

The model***Dima806/traffic\_sign\_detection***is a Vision Transformer (ViT)- based model fine-tuned for traffic sign classification. It was trained on a dataset of various traffic sign images and achieves around 94% accuracy. The model uses ***google/vit-base-patch16-224-in21k*** as its base and is designed to recognize numerous classes such as children crossing, speed limits, U-turn, roundabouts, zebra crossing and many more. The ***dima806/traffic\_sign\_detection*** model tree has some key basic features:

* Model file size: The model weights (model.safetensors) are approximately 343MB
* Performance: The classification report shows strong matrics with many classes achieving precision and recall close to or at 1.0/ indicating high reliabilty for real-world application
* Training Artificats: The respository includes training checkpoints, optimizer states, scheduler files, and prprocessor configurations.
* Deployment: Toy can load and use the model via Hugging face Transformer and Pytorch and it’s licensed under apache 2.0
* Model download: ***https://huggingface.co/dima806/traffic\_sign\_detection***

Other examples of Traffic Sign related datasets include:

1. GTSRB (German Traffic Sign Recognition Benchmark): A widely used dataset for traffic sign classification containing over 50,000 images of traffic signs as well as 43 different classes of traffic signs with various conditions, occlusions and distortions.

2. LISA Traffic Sign Dataset: Contains real-world traffic sign images with variations in lighting and perspectives that helps improve model generalization to different geographical regions.

3. Adversarially Perturbed Versions: Generated using attack methods like FGSM, PGD, and DeepFool.

**3.4.2 DATA PROCESSING**

Preprocessing ensures the dataset is clean and suitable for training ensuring consistency across images from different sources. The following steps are performed:

* Normalization: Pixel values are scaled between 0 and 1 to stabilize training to ensure stable training to the model and prevent numerical instability.
* Resixing and Cropping: All images are sesized to 224x224 pixels which is the standard input size fpr the vision transformer models.
* Data Augmentation: Techniques like rotation, Brightness and contrast adjustments, occlusion and blurring and Gaussian noise are applied to enhance model generalization.
* Attack Injection: Adversarial examples are generated using targeted and untargeted perturbations and added to the dataset.

**3.5 MODEL SELECTION (ViTs) AND TRAINNING**

Core machine learning models used to detect adversarial image attacks, justify the used of Vision Transformers (ViTs) over the normal conventional architectures and outline the complete training pipelines, including hyperparameter tuning.

**3.5.1 VISION TRANSFORMER (ViTs) SELECTION**

Unlike CNNs, ViTs process images as sequences of patches, capturing long-range dependencies and making them more resilient to adversarial perturbations. A pre-trained ViT model is fine-tuned on the adversarial dataset for attack detection.

Traditional deep learning models used in mage classification of CNNs are powerful but have notable weaknesses when it comes to defending against adversarial attacks. ViTs treats an image as a sequence of patches and use self-attention mechanisms to model global dependencies across the entire image. Some of the advantage of selecting the ViTs include

* Global Feature Awareness: Ability to capture long-range relationships between patches.
* Scalability and transferability: A pretrained Vit model can be fine-tuned effectively on smaller domain-specific datasets, (Vit-B/16 or ViT-small) is selected as the backbone of the adversarial detection system.

**3.5.2 TRANSFER LEARNING AND FINE-TUNING**

With a relatively limited size of the adversarial image dataset compared to large-scale datasets like imageNet, Transfer learning is employed to enhance training efficiency and performance. A Vit model pretrained on imageNet-21k is used as the starting point, then during a layer freezing strategy, initial experiments freeze lower layers and only fine-tune higher transformer blocks and the classification head. The final experiment selectively unfreeze more layers to maximize feature learning specific to adversarial patterns.

This process allows the model to adapt learning visual features to the nuances of adversarial perturbation in traffic sign images.

**3.5.3 TRAINNING PROCESS (PIPELINE)**

The model is trained using supervised learning with adversarially perturbed and clean images. The training pipeline is a structure that involves:

* LOSS FUNCTION: Cross-entropy loss for classification between adversarial and non-adversarial images. It is classified mathematically as *L = -{y.log(p) + (1-y) . log(1-p)}*

Where *y* is the true label and *p* is the predicted probability of being adversarial

* OPTIMIZER: AdamW optimizer is used with weight decay for better convergence with a learning rate scheduler for stability. A **cosine learning rate** scheduler with warm-up epochs improves stability during early training.

**3.6 ADVERSARIAL ATTACK DETECTION MODULE**

The Detection Module serves as the core component of the AI-based system that classifies input image as either adversarial or clean (non-adversarial). The trained ViT model is integrated into an adversarial detection module within an autonomous system’s perception pipeline. Given an input image, the model predicts whether it is adversarially manipulated. If detected, mitigation strategies such as reprocessing the image or rejecting the input are applied.

The goal of the module is to serve as protective layer between the image acquisition system of an autonomous platform and its decision making module. The main function of the module indicates:

* Image intake and processing
* Feature extraction and classification
* Real-time flagging of adversarial inputs
* Optional response triggering (e.g., discard, request re-capture, alert system)

**3.6.1 MODULE ANALYSIS AND CLASSIFICATION**

1. Patch embedding: The preprocessed image is divided into fixed-size patches (e.g., 16x16). Each patch is flattened and passed through a linear layer to generate embeddings

2. Self-attention mechanism: The Vit applies multi-head self-attention to learn the relationships between patches across the entire image. This procedure allows the module to identify unusual global patterns caused by adversarial noise.

3. Classification token output: The (CLS) token output is passed through a fully connected layer to produce a probability score:

Score close to 1: Classified as adversarial

Score close to 0: Classified as clean (non-adversarial)

**3.6.2 OUTPUT AND RESPONSE HANDLING**

The architecture allows the detection module to act as a gatekeeper, ensuring that only trusted inmage reach the core AI decision system of the autonomous platform. The system handles the output in one of several ways.

Table 3.2 Output and response handling

|  |  |
| --- | --- |
| **Detection Result** | **System Response** |
| Clean Image | Forward to main perception system (e.g., traffic sign recognition) |
| Adversarial Image | Block or flag image, raise alert; request a new capture or switch backup sensor |

**3.6.3 INTEGRATION WITH AUTONOMOUS SYSTEM (SELF DRIVEN VEHICLE)**

Ensuring real-time efficiency performance and low latency,the detection module can be deployed in several ways;

* On-device: This include edge GPU or onboard AI chip.
* On edge server: In vehicular networks with fast processing.
* As a parallel thread: In the perception pipeline to avoid bottlenecks.

**3.6.4 PERFORMANCE CONSIDERATION**

It is highly important to consider some key factors for the efficient operation of the detection model:

1. Latency: The detection model musy classify image in real-time (e.g., <100 ms per image)

2. Resource usage: Model quantization or pruning may be used to deploy the model on resource-constraint devices.

3. Scalability: The system must scale to handle high input rates in video streams on the system module

4. Robustness: False positive (clean image flagged as adversarial) and false negative ( missed adversarial images) must be minimized.

**3.6.5 LOGGING AND FEEDBACKS**

The detection module maintains a log or flagged images, timestamped and confidence scores which serves multiple purposes for later reference, such purpose include:

* Debugging and audit trails.
* Retraining the newly encountered adversarial patterns.
* Real-time monitoring and system health.

Additionally, a feedback loop can be incorporated where flagged adversarial image are sent to an implemented verification model or human analyst in the automated system.

**3.7 IMPLEMENTATION ENVIRONMENT AND STRATEGY**

* Framework: pytorch or tensorflow (with huggingface transformer for Vit)
* Hardware: trainning is performed on GPUs (Cloud-based GPU)
* Model checkpointing: The best performing model (based on validation F1-score) is saved for evaluation on the test set. The implementation follows a modular approach:

Backend: Developed using Python with TensorFlow/PyTorch for ViT-based detection. Frontend: Web interface or embedded dashboard in an autonomous system for visualization which give direct to input adversarial image.

Deployment: Docker-based containerization for easy integration into existing autonomous vehicle frameworks.

**CHAPTER FOUR**

**DESIGN AND IMPLEMENTATION**

**4.1 INTRODUCTION**

This chapter describes the actual implementation of the AI-Based Detection for Adversarial image traffic sign perturbation. The implementation focuses on integrating traffic sign recognition and object detection using pre-trained machine learning models and deploying the system through a user-friendly web interface built with React and Vite. The main objective of this chapter is to outline the development process, describe the functionality of each component, and provide a clear view of how the system was built and deployed.

The implementation aims to bridge the gap between raw AI inference outputs and an interactive, human readable display by combining modern web development tools with state-of-the-art model inference capabilities in conjunction with convolutional neural network (CNN). The entire system is built using modern frontend technologies (React and Vite) and interacts with machine learning models deployed via the Hugging Face Inference API. The application can perform both traffic sign detection and object recognition in uploaded images. It highlights possible adversarial manipulations through model prediction results, bounding boxes, and confidence scores.

**4.2 DEVELOPMENT ENVIRONMENT AND TOOLS**

The development of the system leveraged a modern front-end stack, ensuring speed, modularity, and maintainability. Below are the tools and libraries used:

Table 4.1 Development environment and tools

|  |  |
| --- | --- |
| **TOOL/LIBRARY** | **PURPOSE** |
| React | Frontend UI framework |
| Vite | Development server and build tool |
| MUI (Material-UI) | React UI component library for styling and layout |
| Styled-components | Custom component-level styling |
| TensorFlow.js + COCO-SSD | Object detection using a pre-trained machine learning model |
| Hugging Face API | Traffic sign detection via cloud-hosted ML interference |
| Chart.js + React ChartJS | Visualization of detection result (if extended) |
| ESLint Code | Linting and consistency enforcement |

**4.3 SYSTEM ARCHITECTURE**

The system architecture of the AI-based detection of adversarial image attacks on autonomous systems is designed as a modular and scalable web-based application that integrates machine learning inference with frontend visualization. The architecture is divided into multiple layers, each responsible for specific tasks: User Interface Layer, Processing Layer, and Inference Layer. This structure ensures clarity, maintainability, and flexibility for future enhancements or deployments.

The system is built using a modular React application that consists of the following major components:

* + 1. USER INTERFACE LAYER (PRESENTATION LAYER)
* Framework: Built using ReactJS with Vite as the bundler for fast development and HMR (Hot Module Replacement)
* Libraries:

- Material-UI (MUI) for consistent design system and layout.

- Styled-components for scoped CSS-in-JS styling.

- Chart.js and react-chartjs-2 for graphical display of performance metrics.

* Features:

- Drag-and-drop image uploader for real-time interaction.

- Visualization of detection results with labeled bounding boxes.

- Real-time loading indicators using Material-UI’s CircularProgress.

* + 1. PROCESSING LAYER (APPLICATION LOGIC)
* State Management: Local state is managed using React’s useState and useEffect hooks.
* Image Handling:

- On image upload, a temporary object URL is generated via URL.createObjectURL() for preview and processing.

- Images are converted into base64 or Blob format before being sent to the inference API.

* Asynchronous Requests:

- Uses fetch API to communicate with the Hugging Face inference service.

- Requests include necessary headers such as Authorization with Bearer tokens.

- Responses are parsed and transformed into a uniform format for further processing.

4.3.3 INFERENCE LAYER (BACKEND ML INTEGRATION)

* Traffic Sign Detection:

- Model: dima806/traffic\_sign\_detection hosted on Hugging Face.

- Task: Object Detection via bounding boxes.

- Output: JSON object containing detected signs with labels, confidence scores, and bounding box coordinates.

* Object Detection:

- Model: @tensorflow-models/coco-ssd used in-browser with @tensorflow/tfjs.

- Task: General-purpose object detection.

- Output: Array of objects each containing class label, confidence, and box coordinates.

* Security:

- Uses environment tokens securely stored in the app (in .env.local for production, not exposed in frontend for best practices).

- Requests are protected by Hugging Face API key authentication.

**4.4 CORE FUNCTIONAL MODULES**

**4.4.1 Image Upload Component**

The ImageUploader.jsx component handles the file selection input and passes the image to the application state.

**A close-up of a computer code

AI-generated content may be incorrect.**

Fig. 4.1 Image Upload component.

On file selection, handleImageUpload generates a temporary URL using URL.createObjectURL(file) for rendering.

This image URL is then passed to detection modules for analysis.

**4.4.2 Object Detection with TensorFlow.js**

The CocoSSDDetector.jsx file integrates TensorFlow’s COCO-SSD model:

**A white background with black text

AI-generated content may be incorrect.**

Fig. 4.2. Object detection with tensorflow.js .

The model loads asynchronously. The selected image is passed into the detector, and an array of predictions is returned, each containing a label, confidence, and bounding box. This can be shown in the confusion matrix below

**4.4.3 Traffic Sign Detection via Hugging Face API**

A utility file utils/trafficSignAPI.js is used to communicate with Hugging Face:

A screenshot of a computer program

AI-generated content may be incorrect.

Fig. 4.3. Traffic sign detection via hugging face.

An access token is generated on the Hugging face API to authenticate the hugging face hub to allow applications to perform based on tokens permission (READ) The image file is sent as a binary blob or base64-encoded data.

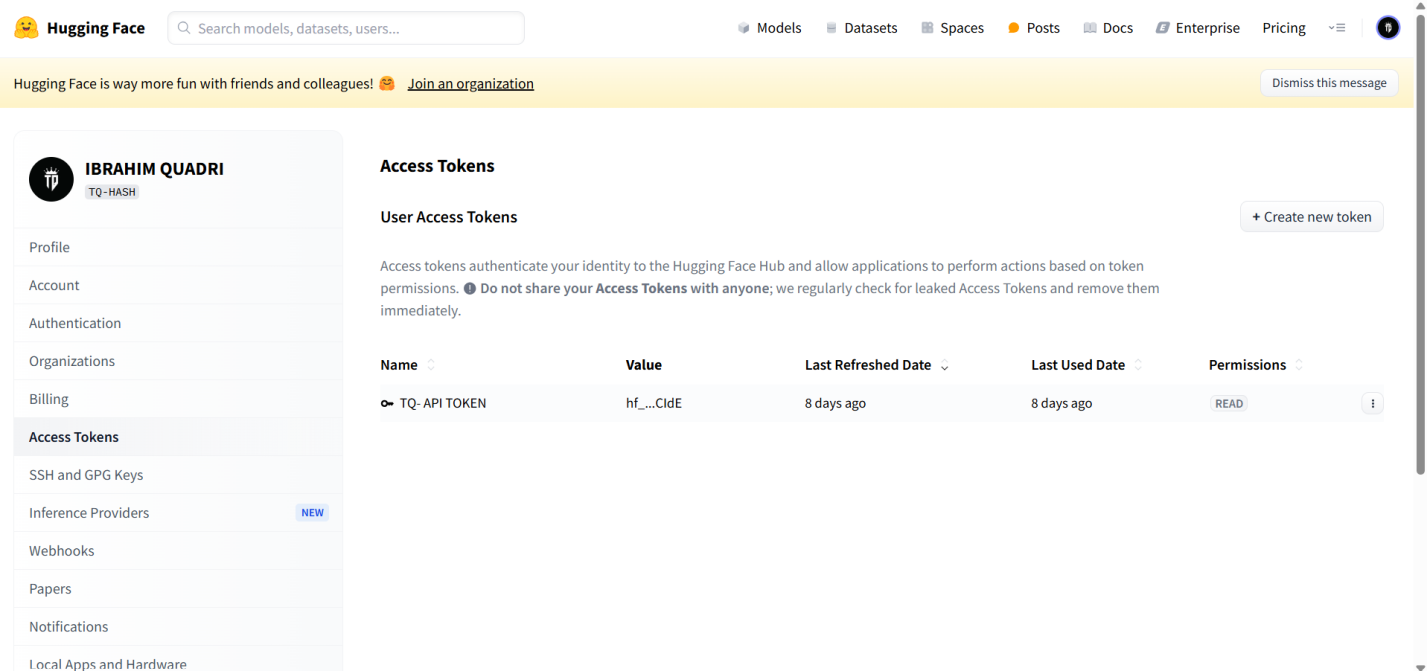


Fig. 4.4. Hugging face access token interface.

The API responds with detected traffic signs, each including label, confidence score, and bounding box. Results are normalized and processed through processTrafficSignResults().

**4.4.4 Confusion matrix**

The model demonstrates high accuracy across most classes, with an overall accuracy of 93.88%. Metrics such as precision, recall, and F1-score are provided for each class. For instance:

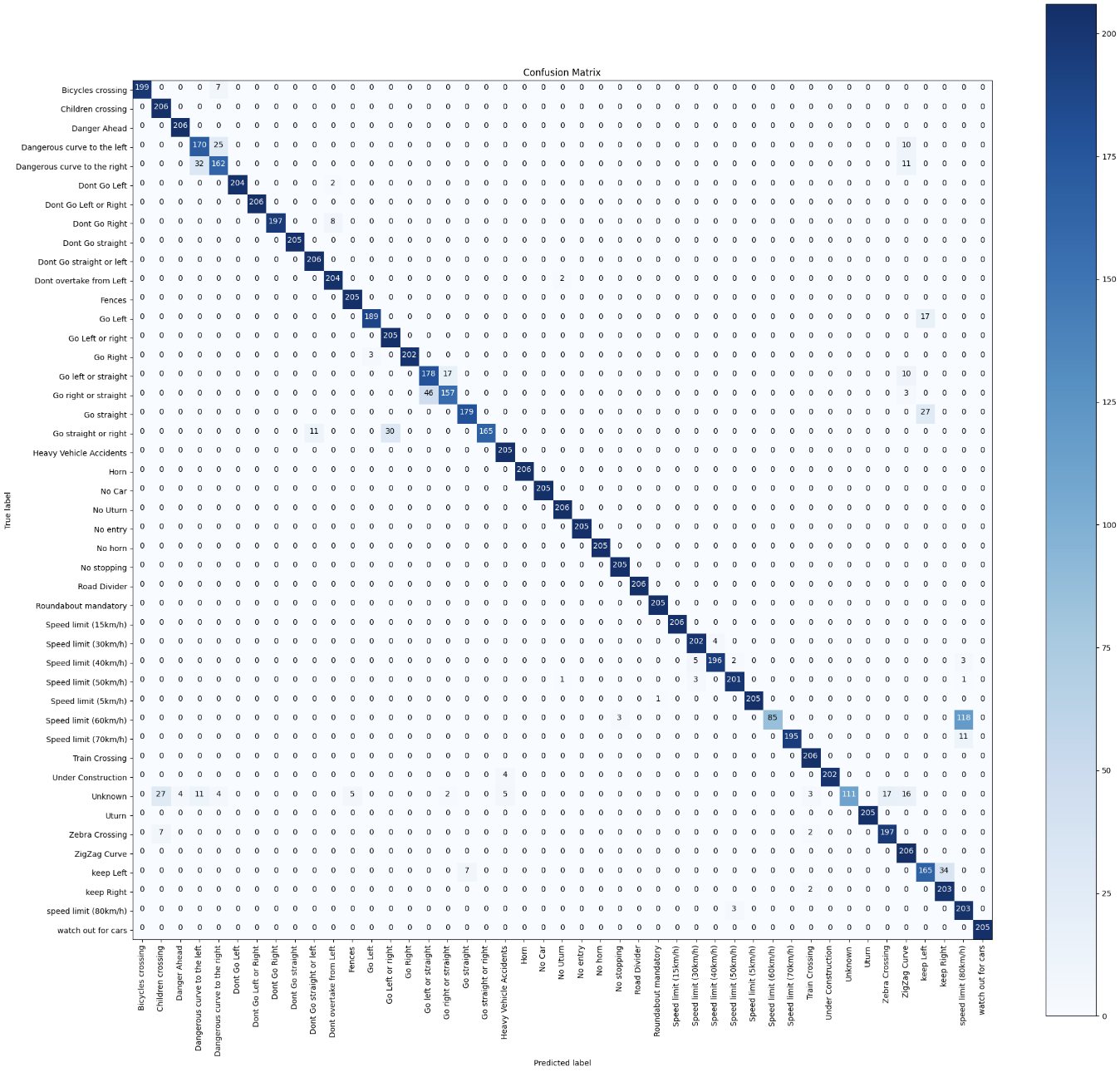


Fig. 4.5. Confusion matrix.

These results are rendered on an overlay positioned above the image.

**4.4.5 CLASSIFICATION REPORT:**

Table 4.2 Traffic signs classification report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PRECESION | RECALL | F1-SCORE | SUPPORT |
| Bicycle crossing | 1.0000 | 0.9660 | 0.9827 | 206 |
| Children crossing | 0.8583 | 1.0000 | 0.9238 | 206 |
| Danger ahead | 0.9810 | 1.0000 | 0.9904 | 206 |
| Dangerous curve to the left | 0.7981 | 0.8293 | 0.8134 | 205 |
| Dangerous curve to the right | 0.8182 | 0.7902 | 0.8040 | 205 |
| Don’t Go Left | 1.0000 | 0.9903 | 0.9951 | 206 |
| Don’t Go Left or Right | 1.0000 | 1.0000 | 1.0000 | 206 |
| Don’t Go Right | 1.0000 | 0.9610 | 0.9801 | 205 |
| Don’t Go straight | 1.0000 | 1.0000 | 1.0000 | 205 |
| Don’t go left | 0.9493 | 1.0000 | 0.9740 | 206 |
| Don’t overtake from Left | 0.9533 | 0.9903 | 0.9714 | 206 |
| Fences | 0.9762 | 1.0000 | 0.9880 | 205 |
| Go Left | 0.9844 | 0.9175 | 0.9497 | 206 |
| Go Left or right | 0.8723 | 1.0000 | 0.9318 | 205 |
| Go Right | 1.0000 | 0.9854 | 0.9926 | 205 |
| Go left or straight | 0.7946 | 0.8683 | 0.8298 | 205 |
| Go right or straight | 0.8920 | 0.7621 | 0.8220 | 206 |
| Go straight | 0.9624 | 0.8689 | 0.9133 | 206 |
| Go straight or right | 1.0000 | 0.8010 | 0.8895 | 206 |
| Heavy Vehicle Accidents | 0.9579 | 1.0000 | 0.9785 | 205 |
| Horn | 1.0000 | 1.0000 | 1.0000 | 206 |
| No Car | 1.0000 | 1.0000 | 1.0000 | 205 |
| No Uturn | 0.9856 | 1.0000 | 0.9928 | 206 |
| No entry | 1.0000 | 1.0000 | 1.0000 | 205 |
| No horn | 1.0000 | 1.0000 | 1.0000 | 205 |
| No stopping | 0.9856 | 1.0000 | 0.9927 | 205 |
| Road Divider | 1.0000 | 1.0000 | 1.0000 | 206 |
| Roundabout mandatory | 0.9951 | 1.0000 | 0.9976 | 205 |
| Speed limit (15km/h) | 1.0000 | 1.0000 | 1.0000 | 206 |
| Speed limit (30km/h) | 0.9619 | 0.9806 | 0.9712 | 206 |
| Speed limit (40km/h) | 0.9800 | 0.9515 | 0.9655 | 206 |
| Speed limit (50km/h) | 0.9757 | 0.9757 | 0.9757 | 206 |
| Speed limit (5km/h) | 1.0000 | 0.9951 | 0.9976 | 206 |
| Speed limit (60km/h) | 1.0000 | 0.4126 | 0.5842 | 206 |
| Speed limit (70km/h) | 1.0000 | 0.9466 | 0.9726 | 206 |
| Train Crossing | 0.9671 | 1.0000 | 0.9833 | 206 |
| Under Construction | 1.0000 | 0.9806 | 0.9902 | 206 |
| Unknown | 1.0000 | 0.5415 | 0.7025 | 205 |
| Uturn | 1.0000 | 1.0000 | 1.0000 | 205 |
| Zebra Crossing | 0.9206 | 0.9563 | 0.9381 | 206 |
| ZigZag Curve | 0.8047 | 1.0000 | 0.8918 | 206 |
| keep Left | 0.7895 | 0.8010 | 0.7952 | 206 |
| keep Right | 0.8565 | 0.9902 | 0.9186 | 205 |
| Speed limit (80km/h) | 0.6042 | 0.9854 | 0.7491 | 206 |
| watch out for cars | 1.0000 | 1.0000 | 1.0000 | 205 |

**4.4.6 Dashboard Detector and Rendering**

The DashboardDetector.jsx component integrates both the traffic sign and COCO-SSD detectors:

* Takes an image URL as a prop.
* Loads and resizes the image.
* Passes it to both detection systems in sequence.
* Results are combined and rendered as annotated boxes and labels on an HTML canvas.
* Bounding boxes are rendered with:

A white rectangular object with a black border

AI-generated content may be incorrect.

Fig. 4.6. Dashboard detector and rendering.

Each detection label is color-coded and rendered near its associated object/traffic sign.

**4.5 STYLING AND LAYOUT**

Styling is handled via a combination of styled-components, MUI, and raw CSS. For instance:

Styling and layout are central to creating a user-friendly and visually consistent interface. In the buildup of the design of the application, styling was implemented using a combination of:

* Material-UI (MUI): A popular React UI framework
* Styled-components: For writing CSS-in-JS
* Raw CSS: app.css and index.css for global/default styles

4.5.1 Material-UI (MUI):

A React-based UI framework that offers a suite of ready-made, accessible components. In this project, MUI was used for layout structure (e.g., AppBar, Box, Container, Paper) and elements like Typography, Buttons, Chip, Divider, and LinearProgress.

4.5.2 Styled-components:

This writes scoped CSS directly in JavaScript using the styled API. This method provides dynamic styling via props, theme-aware responsiveness, and eliminates global style leakage. Custom components like UploadCard, ResultCard, ConfidenceBar, and DetectionCard were styled using styled-components.

4.5.3 Raw CSS (app.css, index.css):

Global styling such as font imports, layout resets, background defaults, and custom scrollbar styling are defined in index.css and app.css. It also includes support for dark/light theme toggles and animations.

**4.6 ESLINT AND CODE QUALITY**

An ESLint configuration (eslint.config.js) is included to ensure adherence to modern JavaScript best practices:

* Enforces React Hooks rules.
* Warns against unused variables.
* Supports JSX and module syntax.
* Prevents anti-patterns in component exports.

This setup promotes clean, maintainable code.

**4.7 BUILD AND DEPLOYMENT**

The application can be started in development mode using vercel.app to deploy the code

This runs the React development server, enabling hot reloading, debugging, and fast iteration during development. The app was deployed using vercel.app

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 4.7. Deployment page.

**4.8 USER INTERFACE**

The interface is designed to be user-friendly, with clear instructions and input fields for users to detect clean images from adversarial images by generating a confidence score after upload.

A screen on the dashboard of a car

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 4.8. User Interface.

**4.9 FUTURE ENHANCEMENTS**

The architectural build up for this project allows future enhancements such as:

* Drawing bounding boxes over detected traffic signs using HTML Canvas or SVG overlays.
* Displaying detection timelines or heatmaps.
* Supporting video input (e.g., dashcam feeds)
* Integrating model switching via a dropdown to compare different Hugging Face models.
* Adding authentication and user-upload history (using Firebase, Supabase, or backend APIs)

The application performs well on desktops, tablets, and mobile devices with minimal layout shift.

**CHAPTER FIVE**

**CONCLUSION, CHALLENGES AND RECOMMENDATION**

**5.1 SUMMARY**

This chapter summarizes findings, conclusions drawn from the system's development and implementation, challenges encountered during the project, and recommendations for future improvements. It encapsulates the essence of the entire research work and provides insights for future endeavors related to AI-based image detection systems for autonomous platforms.

The implementation involved a structured integration of frontend tools (React, MUI), backend API consumption (Hugging Face Inference API), and AI models (for object and traffic sign recognition). The project demonstrates how computer vision can be harnessed to improve the robustness of autonomous systems against manipulated or misleading visual inputs.

**5.2 CONCLUSION**

AI and deep learning-based models are powerful tools in enhancing autonomous systems, particularly in detecting visual cues such as traffic signs and objects on the road. The fusion of pre-trained models, web technologies, and API-based inference provides an efficient pathway for rapid prototyping of intelligent detection systems.

The growing reliance on autonomous systems in domains like transportation, surveillance, and smart infrastructure demands integrating highly accurate and robust visual recognition capabilities. This project set out to develop a system that uses artificial intelligence to detect traffic signs and objects, with the added capability of identifying possible adversarial manipulations in images—a critical threat to the reliability of vision-based autonomous systems. The system successfully employed two major AI components which include the TensorFlow’s COCO-SSD model for general object detection and hugging Face’s dima806/traffic\_sign\_detection model for detecting traffic signs.

These models were integrated into a React-based frontend interface, styled using MUI (Material UI) and styled components, offering a clean and interactive user experience. The user could upload image files, analyzed through the AI models and visual results such as bounding boxes and labels were displayed dynamically.

Thus, enhancing feasibility by developing lightweight, browser-accessible AI-powered image detection applications and incorporating models like coco-ssd and dima806/traffic\_sign\_detection offers impressive performance, although their accuracy can be affected by adversarial image attacks, such as occlusion, noise, or altered pixel values. The integration of Hugging Face models with a React frontend demonstrated that cloud-based AI services are viable for lightweight client applications. Presenting inference results in a user-friendly dashboard increases system transparency and usability.

**5.3 CHALLENGES**

Despite achieving the core objectives of developing a traffic sign and object detection system using AI models, several practical and technical challenges arose during the course of implementation. These challenges provided valuable insights into real-world constraints when integrating machine learning models into web applications, particularly using modern frontend frameworks like React.

The following challenges were encountered during the build up of this project:

1. Model Latency: The Hugging Face API-based model inference sometimes had high latency, especially when used for large image files. This affected real-time usability.

2. Image Upload Limitations: Handling large image sizes or unsupported formats occasionally caused rendering issues or failed inferences.

3. Limited Access to Training Data: Since the models were pre-trained, fine-tuning or evaluating against custom adversarial examples was limited by the lack of access to raw model internals or training data.

4. React Integration Complexity: Integrating TensorFlow.js within a React component structure (especially when combining asynchronous loading, image rendering, and predictions) required state management best practices to avoid memory leaks or inconsistent outputs.

5. Styling and Responsiveness: Ensuring that the interface (using MUI and styled-components) remained clean, responsive, and consistent across screen sizes posed some CSS management challenges, particularly with layered image rendering and bounding boxes.

**5.4 RECOMMENDATIONS**

Based on the findings and challenges encountered, the following recommendations are proposed:

1. Model Optimization: Use model quantization or pruning techniques to reduce latency if deploying offline, also Considering locally hosted models using TensorFlow.js for improved performance and reliability in real-time applications.

2. Adversarial Robustness Testing: Incorporate tools like Foolbox, CleverHans, or Adversarial Robustness Toolbox to test and harden models against known adversarial attacks. Using data augmentation techniques to simulate adversarial examples during training or inference for more robust behavior.

3. Frontend Improvements: Improve UX by adding image previews, error messages, and confidence threshold controls.

4. Scalability and Deployment: Convert the solution into a PWA (Progressive Web App) for offline capability.

5. Dataset Expansion: Collection and usage of custom adversarial traffic sign datasets to better simulate real-world distortions such as rain, blur, or occlusion. Also, Fine-tune models with domain-specific images (e.g., local road signs) for higher accuracy in context-specific applications.

This project has demonstrated how artificial intelligence, combined with modern frontend tools and open-source models, can be used to create an intelligent, responsive system capable of recognizing traffic signs and objects while being aware of potential vulnerabilities. As AI continues to evolve, building secure, explainable, and user-centric systems will be crucial in fostering trust and safety in autonomous applications.

**REFERENCE**

Asha, S., & Vinod, P. (2022). Evaluation of adversarial machine learning tools for securing AI systems. Cluster Computing, 25(1), 503–522. https://doi.org/10.1007/s10586-021-03491-4

Deng, Y., Jin, Q., & Han, Q.-L. (2021). Deep learning-based autonomous driving systems: A survey of attacks and defenses. IEEE Transactions on Industrial Informatics, 17(12), 1–13. https://doi.org/10.1109/TII.2021.3061167

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2022). Adversarial examples in the physical world. In Proceedings of the International Conference on Learning Representations (ICLR). OpenReview.net. https://openreview.net/forum?id=BJm4T4Kgx

Lu, Y., Jia, Y., Wang, J., Li, B., Chai, W., Carin, L., & Velipasalar, S. (2020). Enhancing cross-task black-box transferability of adversarial examples with dispersion reduction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR42600.2020.01024

Monfort, M., Muller, U., Zhang, J., et al. (2016). End-to-end learning for self-driving cars. arXiv preprint arXiv:1604.07316. <https://arxiv.org/abs/1604.07316>

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In Proceedings of the IEEE Symposium on Security and Privacy (S&P) (pp. 582–597). https://doi.org/10.1109/SP.2016.41

Sharma, Y., Yi, J., & Hsieh, C.-J. (2017). ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (pp. 15–26). https://doi.org/10.1145/3128572.3140448

Zhang, H., Yao, Z., & Sakurai, K. (2024). Versatile defense against adversarial attacks on image recognition. arXiv preprint arXiv:2403.08170. <https://arxiv.org/abs/2403.08170>

Zhou, H., Li, W., Kong, Z., Guo, J., Zhang, Y., Yu, B., Zhang, L., & Liu, C. (2020). DeepBillboard: Systematic physical-world testing of autonomous driving systems. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE) (pp. 347–358). https://doi.org/10.1145/3377811.3380376

Zhou, X., Xu, M., Wu, Y., & Zheng, N. (2021). Deep model poisoning attack on federated learning. Future Internet, 13(3), 63. https://doi.org/10.3390/fi13030063

**APPENDIX**

GitHub Repository link: **https://github.com/TQ-HASH/AI-TRAFFIC-**

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.