MULTIPLE ROAD DAMAGE DETECTION USING DEEP LEARNING FRAMEWORK

A PROJECT REPORT

Submitted by

DEVARAJKUMAR R.	**(911720104009)**
PATHMESH E.	**(911720104044)**
TAMIL SELVAN S.	**(911720104078)**
HARIPRASATH S.	**(911720104018)**

in partial fulfillment for the award of the degree

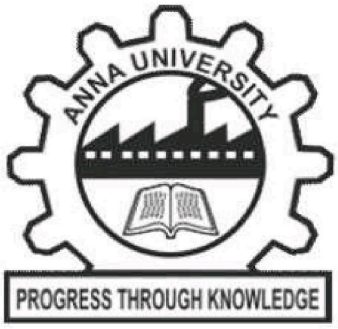
of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

MOUNT ZION COLLEGE OF ENGINEERING AND TECHNOLOGY



ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY, CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled

Certified that this project titled "MULTIPLE ROAD DAMAGE DETECTION USING DEEP LEARNING FRAMEWORK" is a bonafide work of **PATHMESH E (REG.No: 911720104044), DEVARAJKUMAR S (REG.NO: 911720104009), TAMILSEVAN S (REG.NO: 911720104078), HARIPRASATH S (REG.NO: 911720104018)** who carried out the work under my supervision, for the partial fulfillment of the requirements for the award of the degree of *Bachelor of Engineering* in *Computer Science & Engineering*. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion.

SIGNATURE

Mrs. D. ELAVARASI, M.E., Ph.D*,

HEAD OF THE DEPARTMENT

Assistant Professor,
Department of Computer Science
and Engineering,
Mount Zion College of
Engineering &Technology,
Pilivalam Post, Thirumayam TK.,
Pudukkottai – 622 507 .

SIGNATURE

Mr. K.GURUNATHAN, M.E, Ph.D.,

SUPERVISOR

Assistant Professor,
Department of Computer Science
and Engineering,
Mount Zion College of
Engineering &Technology,
Pilivalam Post, Thirumayam
TK., Pudukkottai – 622 507 .

Submitted for the Viva-Voce Examination held on __

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I hereby declare that the work entitled "**CAMPUS CAREER HUB**" is submitted in partial fulfillment of the requirement for the award of the degree in B.E., Anna University Chennai, is a record of the work carried out by me during academic year 2023 – 2024 under the supervision and guidance of Mrs. ELAVARASI. D, H.O.D, Department of Computer Science and Engineering, Mount Zion College of Engineering and Technology. The extent and source of information are derived from the existing literature and have been indicated through the dissertation at the appropriate places. The master embodied in this work is original and has not been submitted for the award of any other degree or diploma, either in this or any other university.

Signature of the candidate,

DEVARAJKUMAR R

PATHMESH E

TAMILSELVAN S

HARIPRASATH S

ACKNOWLEDGEMENT

From my deepest soul, I thank God, the Almighty for helping me complete my project work.

I am wholeheartedly grateful to **Mrs. Florence Jayabarathan, M.A. M.Ed ChairPerson,** Mount Zion College of Engineering & Technology, for providing me with all the facilities which helped me to do my project work.

I would like to express my sincere thanks to **Prof. Jayson K. Jayabarathan, M.Tech., Ph.D., Director,** Mount Zion College of Engineering & Technology, for his technical expertise for successful completion of my project work.

With a deep sense of gratitude, I record my thanks to **Dr. P. Balamurugan, M.E., Ph.D., Principal,** Mount Zion College of Engineering & Technology, for all his support throughout my work.

I am vastly indebted to **Mrs. Elavarasi D, M.E., Ph.D.*, Head of the Department, Department of Computer Science and Engineering,** for her immense encouragement, support and guidance to complete my project.

I express my sincere thanks to my guide **Mr. Gurunathan K,M.E.,Ph.D*, Assistant Professor, Department of Computer Science and Engineering** for her help to do my project.

I also would like to thank the Project Coordinator, **Mr. Sathish Kumar G, M.E., Assistant Professor, Department of Computer Science and Engineering** for his constant encouragement through my work.

I would also like to thank all the staff members of the CSE department for their kind cooperation. Also, I express a sense of gratitude and love to my friends and my family for their support and strength that helped me carry out my project successfully.

ABSTRACT

Road infrastructure plays a crucial role in transportation systems, ensuring the safe and efficient movement of people and goods. However, the deterioration of roads over time due to various factors, such as weather conditions and heavy traffic, poses significant challenges for maintenance and safety. This research proposes a novel approach to road damage detection using Convolutional Neural Networks (CNNs). CNNs have shown remarkable success in various computer vision tasks, making them a promising choice for automated road damage detection. The objective of this study is to leverage the power of deep learning and computer vision techniques to develop an efficient and accurate system for detecting road damage from images. Our methodology involves the collection of a diverse dataset of road images containing various types of damage, including potholes, cracks, and road surface degradation. We preprocess the dataset to enhance image quality and annotate it for training and evaluation. A custom CNN architecture is designed and trained using this dataset to recognize and classify different types of road damage. The trained model is evaluated using a separate validation dataset to assess its performance in terms of accuracy, precision, recall, and F1 score. Additionally, we explore the model's ability to generalize to unseen road damage scenarios by conducting testing on real-world images captured under varying conditions. The results indicate that our CNN-based road damage detection system achieves high accuracy in identifying and classifying road damage types. This system can be integrated into existing infrastructure management systems, enabling timely and cost-effective road maintenance. Moreover, it contributes to improved road safety by identifying potential hazards before they lead to accidents.

LIST OF ABBREVIATIONS

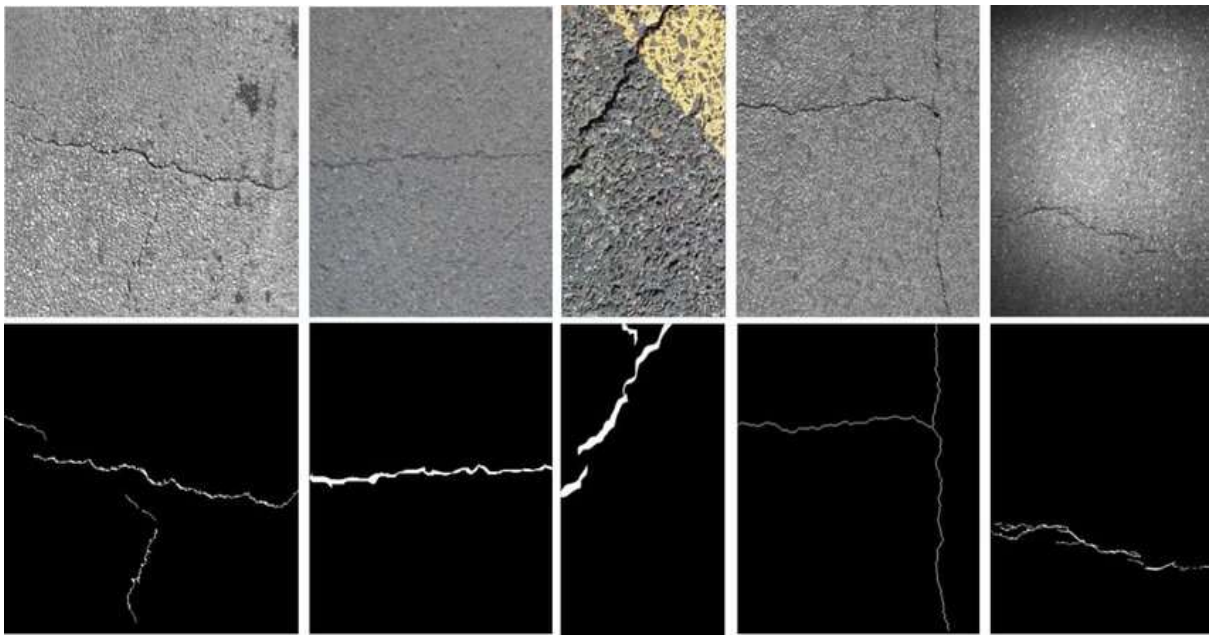
1	RCM	Revenue Cycle Management
2	LiDAR	Light Detection and Ranging
3	GPR	Ground-pentrating radar
4	DL	Deep Learning
5	IRI	Information Resources,Inc
6	RNN	Recurrent Neura Networks
7	DL	Deep Learning
8	GPL	General Public Licence
9	GAN	Generative Adversarial Networks
10	CPP	Cracks Per Patch
11	IGCNN -RCD	Intelligent Graph Convolution Neural Network for Road Crack Detection

CHAPTER 1

1. INTRODUCTION

1.1 OBJECTIVE

Road crack detection is a crucial aspect of infrastructure maintenance and safety. As roads deteriorate over time due to factors like weather, traffic, and general wear and tear, cracks can form, posing hazards to both vehicles and pedestrians. The timely identification and repair of these cracks are essential to ensure the longevity and reliability of road networks. In recent years, technological advancements, particularly in computer vision and image processing, have paved the way for innovative solutions in road crack detection. Automated systems equipped with sophisticated algorithms can analyse images of road surfaces, identifying and classifying various types of cracks with high accuracy. These systems offer a faster and more efficient alternative to traditional manual inspections, enabling authorities to prioritize and address maintenance needs promptly. The implementation of road crack detection technology not only enhances safety but also contributes to cost-effectiveness by preventing minor issues from escalating into major repairs. Moreover, the data collected through these systems can be utilized for predictive maintenance, allowing authorities to anticipate and address potential road issues before they become critical. This introduction sets the stage for exploring the importance and applications of road crack detection, emphasizing its role in preserving infrastructure integrity, enhancing safety, and optimizing maintenance practices in our modern urban environments.



1.2 OVERVIEW

Road infrastructure is susceptible to the formation of cracks over time, leading to safety hazards, increased maintenance expenditures, and potential disruptions to traffic. The manual inspection of roads for crack detection is time-consuming, resource-intensive, and often results in delayed identification of critical issues. This hinders the ability to proactively address road defects, increasing the risk of accidents and necessitating more extensive and costly repairs. Moreover, traditional methods of road inspection may not be able to keep pace with the rapid expansion of road networks and the growing demand for timely maintenance. The lack of efficient and automated crack detection systems exacerbates the challenges associated with preserving road safety, minimizing repair costs, and ensuring the longevity of infrastructure. The problem encompasses the need for a comprehensive and technologically advanced solution that can automate the process of road crack detection. This solution should not only enhance the speed and accuracy of crack identification but also enable predictive maintenance strategies, thereby mitigating safety risks, optimizing resource utilization, and contributing to the overall sustainability of road infrastructure.

Road damage analysis involves assessing the condition of road surfaces to identify and categorize various types of damage such as potholes, cracks, rutting, and surface distress. This analysis is critical for infrastructure maintenance and ensures road safety. Several methods, including computer vision and machine learning techniques, are employed for efficient and automated road damage analysis. One approach is the utilization of image processing and computer vision algorithms to analyze images or videos captured by cameras mounted on vehicles or drones. Convolutional Neural Networks (CNNs) are commonly employed for image-based road damage detection. These models are trained on labeled datasets to recognize patterns and features associated with different types of damage. LiDAR (Light Detection and Ranging) technology is another valuable tool for road damage analysis. LiDAR sensors can provide detailed 3D information about the road surface, allowing for precise measurement and characterization of damages. Additionally, data from road condition surveys, collected by specialized vehicles equipped with sensors, can be analyzed to assess the structural integrity of roads. Machine learning models trained on this data can predict potential road damage and assist in prioritizing maintenance efforts. Overall, road damage analysis integrates advanced technologies to automate the detection, classification, and assessment of damages, facilitating effective and timely maintenance strategies for transportation infrastructure.The objectives for road crack detection can be multifaceted, aiming to enhance safety, streamline maintenance processes, and extend the lifespan of infrastructure. Here are several key objectives for implementing road crack detection systems:

- Early Detection and Prevention: Identify cracks at an early stage to prevent them from developing into more significant structural issues. Minimize the risk of accidents and damage to vehicles by addressing road defects promptly.
- Infrastructure Safety: Enhance overall road safety by identifying and repairing cracks that could compromise the integrity of the road surface. Reduce the likelihood of accidents and injuries caused by uneven or damaged road conditions.
- Cost-Efficient Maintenance: Implement a proactive approach to maintenance, addressing minor cracks before they escalate into major repairs. Optimize budget allocation by prioritizing repairs based on the severity and location of detected cracks.
- Data-Driven Decision-Making: Collect and analyze data on road conditions to make informed decisions about maintenance priorities. Utilize historical data for predictive maintenance, identifying trends and patterns in road deterioration.
- Efficient Resource Allocation: Streamline the allocation of human and financial resources by focusing efforts on areas with the most critical maintenance needs. Minimize disruption to traffic flow by efficiently planning and executing maintenance activities.

1.3 LITERATURE SURVEY

1.3.1 TITLE: AN ACTIVE LEARNING METHOD WITH DIFFICULTY LEARNING MECHANISM FOR CRACK DETECTION

JOURNAL AND YEAR: RESEARCH GATE, 2022

AUTHOR: JIANGPENG SHU, ET.AL

This paper the attempted of active learning is to maximize a model's performance while annotating the fewest samples possible. (Ren et al. 2020) Active learning can be used as a method to reduce the cost of samples annotations, while retaining the powerful learning capabilities of deep learning. In the machine learning, both semi-supervised learning and active learning utilize the labelled and unlabelled samples on autonomous label engineering. Generally, semi-supervised learning does not require manual participation, and it can automatically label the unlabeled samples through a benchmark classifier with a certain classification accuracy. One of the characteristics of active learning, which is different from semi-supervised learning, is that the selected high-value samples need to be marked manually and accurately. Semisupervised learning replaces manual labeling with automatic or semi-automatic labeling by computer. Although the labeling cost is effectively reduced, the labeling results depend on the classification accuracy of the benchmark classifier trained with some labeled samples, so the labeling results cannot be guaranteed to be completely correct. In contrast, active learning selects samples manually and does not introduce

error class labels. In this study, a difficulty-learning active learning method to select the most informative crack images is proposed, aiming at promoting crack segmentation dataset construction. An acquisition function that is combined with difficulty and traditional uncertainty maps is utilized to measure the informativeness of crack images in our method.

1.3.2 TITLE: RESEARCH ON PAVEMENT CRACK DETECTION ALGORITHM BASED ON DEEP RESIDUAL UNET NEURAL NETWORK

JOURNAL AND YEAR: IOPSCIENCE, 2022

AUTHOR: YALONG YANG, ET.AL

With the rapid development of China's economy, China's highway mileage is also increasing rapidly. Up to now, the total mileage of highways has exceeded 5 million kilometers, which puts forward higher requirements for highway maintenance. Cracks are the initial symptoms of pavement damage. In order to ensure traffic safety, pavement cracks should be found and remedied in time. At present, the detection of pavement crack mainly depends on manual inspection in practice, which is not only expensive inefficient and labor-intensive, but also has subjective. It may lead to missed detection or wrong classification due to bad physical condition or lack of concentration. Therefore, how to automatically detect pavement crack accuratelly has become one of the current research hotspots. Aiming at the segmentation inaccuracy caused by complex background and fuzzy crack edge in asphalt pavement image, an improved pavement crack segmentation method based on deep convolutional neural network model is proposed in this paper, which combines the idea of semantic segmentation with image detection. Experimental results show the proposed method is superior to the classical threshold segmentation iterative segmentation method and the traditional semantic segmentation method UNet in terms of F1 score,Precision,and Recall, and has better segmentation effect. The proposed method is expected to solve the problem of crack detection in complex asphalt pavement images. Therefore, the methods presented in this paper can be applied to crack detection of highway pavement to a certain extent, and better assist the maintenance of highway traffic safety.

1.3.3 TITLE: TOWARDS ROBOTIC MARBLE RESIN APPLICATION: CRACK DETECTION ON MARBLE USING DEEP LEARNING

JOURNAL AND YEAR: MDPI JOURNAL, 2022

AUTHOR: ELENI VROCHIDOU, ET.AL

This paper provided dense pixel-level information useful to fully understand a scene. However, most of the backbones used in segmentation models for feature extraction resulting from

pretrained models may lead to poor performance, especially in small categories, such as marble cracks, due to the lack of sufficient spatial information. Therefore, efficient combinations of model architectures need to be investigated for the problem under study. In this work, 4 deep convolutional neural network models are fully trained (all layers) and 28 backbone architectures for feature extraction are tested for marble crack semantic segmentation. The reason to fully train the models is due to the aforementioned weakness of pretrained models to extract meaningful spatial features from new image samples for the problem under study. Furthermore, to strengthen this decision, a pretrained baseline model is also examined for comparative reasons. Cracks can occur on different surfaces such as buildings, roads, aircrafts, etc. The manual inspection of cracks is time-consuming and prone to human error. Machine vision has been used for decades to detect defects in materials in production lines. However, the detection or segmentation of cracks on a randomly textured surface, such as marble, has not been sufficiently investigated. This work provides an up-to-date systematic and exhaustive study on marble crack segmentation with color images based on deep learning (DL) techniques.

1.3.4 TITLE: CRACK DETECTION IN CONCRETE STRUCTURES USING DEEP LEARNING

JOURNAL AND YEAR: MDPI, 2022

AUTHOR: VAUGHN PETER GOLDING, ET.AL

This paper investigated the effects of preprocessing images on the performance of DL crack detection using a dataset of 40,000 images. The results depicted that using a pretrained model with RGB weights and grayscale images does not affect the performance of a CNN model for detecting cracks in the concrete structure. The other IP methods (thresholding and edge detection) reduced the performance. The grayscale was found to be promising in reducing the noise of the image without removing relevant features. The study used the Keras Python package and pretrained VGG16 to develop the CNN. The original image dataset was converted using the SciKit Image Python package into four sets to compare: RGB (control), luminance (grayscale), Otsu method (thresholding), and Sobel filter (edge detection). This study demonstrated that colour is not a relevant feature in DL crack detection. This was promising, as colour images are larger, and decreasing image data size could increase processing speed and decrease the data size needed for storage. These results may be misleading due to the RGB weights in the pretrained model. The weights for the pretrained models are only available in three-channel (RGB). This was performed due to a lack of time and the greater knowledge needed to execute. Further research should either train the models from scratch on their respective images or obtain one-channel (grayscale)

pretrained weights. Infrastructure, such as buildings, bridges, pavement, etc., needs to be examined periodically to maintain its reliability and structural health.

1.3.5 TITLE: AUTOMATIC PAVEMENT CRACK DETECTION FUSING ATTENTION MECHANISM

JOURNAL AND YEAR: MDPI, 2022

AUTHOR: JUNHUA REN, ET.AL

This paper analysed the cracks on the pavement are a significant sign of potential damage and degradation in the performance and functionality of pavements. Generally, cracks of the pavement can be caused by heavy traffic, drastic temperature change, reflection from the base layers, and so on. These cracks have a negative impact on the structure of the pavement, significantly decreasing the performance of the pavements. With cracks developing, the whole structure of the road can be influenced. The safety and service life of the road can be decreased to a certain degree. To curb the development of cracks, a frequent inspection is needed. By collecting various types of data for pavement conditions, the corresponding strategies and in-depth analysis can be made. According to the analysis results, timely and appropriate maintenance can be employed to repair the pavement and prevent its failure at an early stage of crack development. In this way, as for the pavement, the service life can be prolonged, and the performance and functionality can be maintained in a good condition. As most previous studies have stated, collecting and analyzing the images of cracks is a primary way to realize the detection and classification of pavement cracks. Generally, the pavement crack detection methods can be divided into two types based on whether the deep learning method is applied. They are the traditional methods and deep learning methods. However, in Mosaic, four pictures are randomly selected to join together through random rotation, random scaling, and random distribution. In sum, there were three advantages to employing Mosaic. Firstly, the background of the detected object becomes rich. Secondly, a better model can be more easily trained with fewer GPU resources due to the combination of four pictures.

1.3.6 TITLE: CLOUD-BASED COLLABORATIVE ROAD-DAMAGE MONITORING WITH DEEP LEARNING AND SMARTPHONES

JOURNAL AND YEAR: MDPI, 2022

AUTHOR: AKSHATHA RAMESH

This paper focuses on the problem of monitoring road-surface conditions for detecting possible road damage. A deep-learning and cloud-based collaborative method is proposed. The method utilizes

both vehicle-motion data and vision data collected by cell phones to achieve reliable and accurate detection while being cost-effective. Deep-learning-based techniques including YOLOv5 and LSTM were trained for this task. The cloud fuses detecting results from various vehicles to generate a map with credible road-damage information for drivers. The performance of the proposed method was proven in real world experiments. However, the work can still be improved in the following aspects: The current dataset we use to train the deep-learning models can be expanded to be more comprehensive. In the future, we will collect data from more weather and lighting conditions and vehicle types. The current severity-estimation method is mainly based on motion data. In our future work, we will combine the current estimation method with information from vision data to achieve the estimation of the 3-D dimensions of road damage. In addition, the current method does not output the pavement-quality index, which is a standard index for evaluating road-surface conditions. Modern cellphones are packed with advanced sensors, including a magnetometer, accelerometer, gyro, and GPS, which enable them to measure precise acceleration and speed. When vehicles are driven on roads with different surface conditions, cellphones will receive acceleration stimuli with different patterns and amplitudes, and such different patterns can be used by machine-learning models to detect and classify road damage such as potholes and cracks.

1.3.7 TITLE: ROAD CONDITION MONITORING USING SMART SENSING AND ARTIFICIAL INTELLIGENCE: A REVIEW

JOURNAL AND YEAR: MDPI, 2022

AUTHOR: ESHTA RANYAL

In this paper, the existing research work in RCM using next-generation sensors and AI methodologies was extensively reviewed and compared. The existing approaches are evaluated based on the data acquisition platforms under various AI approaches, especially DL for classification, segmentation and object detection. Data acquisition systems, which are a combination of non-intrusive sensors and their platforms, are at the center of the RCM system and involve the collection of 1D data, 2D visual data or 3D depth data. Every platform has its own advantages and limitations and complements each other's usability. For the tasks involving a simple classification of distresses, RGB sensors are a good option, while for a detailed study involving the various characteristics of distresses, LiDAR, laser, thermal or GPR sensors can be used. It is also necessary to appreciate the value and importance of data, along with the need to understand with clarity the definition of data. Whether the problem deals with classification, detection, or localization in RCM, at the core of every DL vision algorithm is a large collection of labeled images. Thus, another important step is the data annotation of collected data as ground truths, a crucial step in determining the accuracy of any DL model. The results obtained, in comparison with reference data from highly specialized equipment, confirm the proposed RCT solution established an algorithm to calculate the IRI from the acceleration values obtained using smartphone sensors. The algorithm identified the physical parameters of a quarter actual vehicle model and established a relation between the acceleration, IRI and the profile elevation. It was observed that a consideration of the dynamic characteristics of vehicles improved the proposed method's measurement accuracy.

1.3.8 TITLE: ROAD CRACK DETECTION USING QUATERNION NEURAL NETWORKS

JOURNAL AND YEAR: IEEE, 2022

AUTHOR: AGGELOS KATSALIROS

Road crack detection is a complex task which involves detecting irregularities and damage in road surfaces such as concrete, cement, pavement or asphalt. It is a multi-faceted task which also requires classification and segmentation in order to produce accurate results. Due to their nature, cracks in surfaces involve many low-level features and visual patterns and are thus hard to detect and classify. When designing accurate and robust models for road crack detection, one needs to take into account additional factors that make this process challenging. Such factors include sub-par

image quality which often appears in the form of variances in illumination, low resolution, or blurring, as well as visual noise, for example from stains, shadows, reflections, or litter on the road surface. Applications for road crack detection can be found in real life problems, such as automatic car driving safety, automated road inspection and also in ensuring the quality of transport services. In this work, we compare the results of image classification and segmentation CNN architectures with their respective quaternionic counterparts in the task of road crack detection. Specifically, we replace standard layers with quaternion-valued versions and compare the accuracy, precision and number of parameters required by the new models. A reduced-data training regime is devised, in which new datasets are produced by sampling from a large source dataset of road crack images to assess network performance in conditions where training data is not readily available.

1.3.9 TITLE: A NOVEL DATA AUGMENTATION METHOD FOR IMPROVED VISUAL CRACK DETECTION USING GENERATIVE ADVERSARIAL NETWORKS

JOURNAL AND YEAR: IEEE, 2023

AUTHOR: EFSTATHIOS BRANIKAS

Structural health inspection monitoring and defect detection is a field that is suffering from data imbalances when quantifying this problem for learning-based methods while acquiring fully annotated datasets is extremely time consuming. It would be desirable to tackle these challenges in a manner that can both generalize across different data modalities and require minimal manual work. Therefore, in this paper an approach to this end is presented, by modifying and implementing a well-established generative adversarial network for a cyclic image-to-image translation to perform domain transfer in two domains. In this way an augmentation is achieved on both domains and in the case where an existing annotation is available for only one dataset, the new generated data on the other domain can share this annotation, creating an artificial annotated dataset. Through extensive experiments conducted in three different test cases, it is shown that the proposed augmentation framework alleviates the challenges mentioned above and improves the crack segmentation process, providing state-of-the-art results in some cases. As one specific segmentation neural network is deployed to test the effectiveness of the proposed augmentation framework, in future work it could be beneficial to further investigate how the GAN topology is associated with the detection results of other segmentation algorithms.

1.3.10 TITLE: TOPO-LOSS FOR CONTINUITY-PRESERVING CRACK DETECTION USING DEEP LEARNING

JOURNAL AND YEAR: ELSEVIER, 2022

In this paper, we overcome both of these shortcomings. We first make use of the topological loss (TOPO), which specifically penalizes discontinuities. For the assessment of continuity, we introduce a new metric that counts the number of cracks in an input image. We next show that this loss can accommodate coarse labels, significantly reducing the labeling effort needed to deploy our system. We present experiments on two datasets, in which we compare the performance of TernausNet, a standard deep network architecture previously used for crack detection, that has been trained with different loss functions. We specifically focus on the capacity of the network to correctly represent the topology of cracks. To compare the topology of the predicted and annotated cracks, we introduce a new dedicated metric that we call Cracks Per Patch (CPP). Both qualitative and quantitative results show that the use of TOPO significantly improves crack continuity while reducing false positives. We make both our code and the new dataset publicly available. The main drawback of using the topological loss function is a possible loss of precision in crack localization. Though loss enforces crack continuity, it relaxes the requirement of a perfect coincidence between the predicted and the annotated cracks. The estimated crack centers may therefore be slightly offset from the annotated ones, resulting in a lower accuracy in pixel classification, as highlighted in Section 4. Moreover, TOPO is designed to segment thin elongated structures, which may misrepresent the crack width. Additional post-processing may be applied if precise width estimates are desired.

CHAPTER 2

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The IGCNN-RCD (Intelligent Graph Convolution Neural Network for Road Crack Detection) proposes a significant enhancement to the existing road crack detection system by introducing a novel approach to feature extraction and learning. In the conventional system, features are likely extracted using traditional image processing or standard deep learning methods, and the representation of these features is typically in vector or matrix form. The IGCNN-RCD, on the other hand, incorporates the Scale-Invariant Feature Transform (SIFT) algorithm for feature extraction, offering scale and rotation-invariant features. This integration enables the system to capture more distinctive key points and descriptors, providing a richer representation of the road surface, particularly in the context of intricate crack patterns. Moreover, the IGCNN-RCD introduces a graph-based representation, constructing graphs to depict relationships between the key points extracted by the SIFT algorithm. This graph-based approach enables the system to capture spatial relationships and similarities between features, fostering a more nuanced understanding of the road surface characteristics. The final layer of enhancement involves the incorporation of a Graph Convolutional Neural Network (GCNN) for learning from the graph representation of road images. GCNNs excel at learning from complex relationships within graph structures, enabling the system to discern various defects, including road cracks, with improved accuracy and efficiency. In essence, the IGCNN-RCD builds upon the foundations of the existing system, offering a more sophisticated and effective framework for road crack detection through the synergistic integration of SIFT-based feature extraction and graph-based learning.

2.1.1 DISADVANTAGES

- Does not support multiple road cracks
- Time complexity can be high
- Machine learning based approach (Feature extraction and learning)
- Computational complexity problem
- Need large trained datasets

2.2 PROPOSED SYSTEM

The proposed system for road damage detection leverages the capabilities of Convolutional Neural Networks (CNNs) to enhance the precision and effectiveness of identifying various types of road damages. The first step involves the collection of a comprehensive dataset comprising diverse road images, encompassing different damage scenarios such as cracks, potholes, and surface deterioration. This dataset is crucial for training the CNN to recognize and classify various forms of road damage accurately. Subsequently, a series of preprocessing steps are applied to the collected images, ensuring uniformity and optimizing model performance. This preprocessing may include resizing images, normalizing pixel values, and employing data augmentation techniques to enhance the diversity of the dataset. The core of the proposed system lies in the design of a dedicated CNN architecture tailored for road damage detection. The CNN architecture typically comprises convolutional layers responsible for feature extraction, pooling layers for down-sampling, and fully connected layers for classification. The convolutional layers enable the model to automatically learn and extract relevant features from the input images, while pooling layers contribute to spatial down-sampling, reducing computational complexity. The fully connected layers then process the extracted features to make predictions regarding the presence and nature of road damages. By implementing this integrated approach, the proposed system aims to offer a robust solution for road damage detection, capable of generalizing well across diverse road conditions and effectively identifying various types of damages through the power of convolutional neural networks.

2.2.1 ADVANTAGES

- Multiple road cracks are analysed
- Accuracy can be improved
- Reduce the time complexity and computational complexity

CHAPTER 3

3. SYSTEM SPECIFICATION

3.1 HARDWARE REQUIREMENTS

- Processor : Intel core processor 2.6.0 GHZ
- RAM : 4 GB
- Hard disk : 160 GB
- Keyboard : Standard keyboard
- Monitor : 15-inch colour monitor

3.2 SOFTWARE REQUIREMENTS

- Server Side : Python 3.7.4(64-bit) or (32-bit)
- Client Side : HTML, CSS, Bootstrap
- IDE : Flask 1.1.1
- Back end : MySQL 5.
- Server : WampServer 2i
- OS : Windows 10 64 –bit

CHAPTER 4

4. SOFTWARE DESCRIPTION

4.1 PYTHON

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages due to its simplicity, readability, and versatility. One of the key features of Python is its easy-to-learn syntax, which makes it accessible to both novice and experienced programmers. It has a large standard library that provides a wide range of modules for tasks such as file I/O, networking, regular expressions, and more. Python also has a large and active community of developers who contribute to open-source libraries and packages that extend its capabilities. Python is an interpreted language, which means that it is executed line-by-line by an interpreter rather than compiled into machine code like C or C++. This allows for rapid development and testing, as well as easier debugging and maintenance of code. Python is used for a variety of applications, including web development frameworks such as Django and Flask, scientific computing libraries such as NumPy and Pandas, and machine learning libraries such as TensorFlow and PyTorch. It is also commonly used for scripting and automation tasks due to its ease of use and readability. Overall, Python is a powerful and versatile programming language that is widely used in a variety of domains due to its simplicity, ease of use, and active community.

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering

choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture."Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one —and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. [When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar. A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

Python also has a large and active community of developers who contribute to a wide range of open-source libraries and tools, making it easy to find and use pre-built code to solve complex problems.

Python has a wide range of applications, including:

Data Science: Python is one of the most popular languages for data science, thanks to libraries like NumPy, Pandas, and Matplotlib that make it easy to manipulate and visualize data.

Machine Learning: Python is also widely used in machine learning and artificial intelligence, with libraries like TensorFlow, Keras, and Scikit-learn that provide powerful tools for building and training machine learning models.

Web Development: Python is commonly used in web development, with frameworks like Django and Flask that make it easy to build web applications and APIs.

Scientific Computing: Python is used extensively in scientific computing, with libraries like SciPy and SymPy that provide powerful tools for numerical analysis and symbolic mathematics.

In addition to its versatility and ease of use, Python is also known for its portability and compatibility. Python code can be run on a wide range of platforms, including Windows, macOS, and Linux, and it can be integrated with other languages like C and Java.

Overall, Python is a powerful and versatile programming language that is well-suited for a wide range of applications, from data science and machine learning to web development and scientific computing. Its simplicity, readability, and large community of developers make it an ideal choice for beginners and experts alike.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.

2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

One of the strengths of Python is its rich ecosystem of third-party libraries and tools. These libraries provide a wide range of functionality, from scientific computing and data analysis to web development and machine learning. Some popular Python libraries and frameworks include:

NumPy: a library for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays.

Pandas: a library for data manipulation and analysis in Python, providing support for reading and writing data in a variety of formats, as well as powerful tools for manipulating and analyzing data.

Matplotlib: a plotting library for Python that provides a variety of visualization tools, including line plots, scatter plots, bar plots, and more.

TensorFlow: an open-source machine learning library for Python that provides a variety of tools and algorithms for building and training machine learning models.

Django: a popular web framework for Python that provides a full-stack framework for building web applications, with support for everything from URL routing to user authentication and database integration.

Python's popularity has also led to a large and active community of developers who contribute to open-source projects and share code and resources online. This community provides a wealth of resources for learning Python, including tutorials, online courses, and forums for asking and answering questions.

Overall, Python is a versatile and powerful programming language that is well-suited for a wide range of applications. Its simplicity, flexibility, and wide range of libra

4.2 TENSORFLOW LIBARIES IN PYTHON

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It is one of the most popular libraries for building and training machine learning models, especially deep neural networks. TensorFlow allows developers to build complex models with ease, including image and speech recognition, natural language processing, and more. One of the key features of TensorFlow is its ability to handle large-scale datasets and complex computations, making it suitable for training deep neural networks. It allows for parallelization of computations across multiple CPUs or GPUs, allowing for faster training times. TensorFlow also provides a high-level API called Keras that simplifies the process of building and training models. TensorFlow offers a wide range of tools and libraries that make it easy to integrate with other Python libraries and frameworks. It has built-in support for data preprocessing and visualization, making it easy to

prepare data for training and analyze model performance. One of the major advantages of TensorFlow is its ability to deploy models to a variety of platforms, including mobile devices and the web.

Graph-based computation: TensorFlow uses a graph-based computation model, which allows for efficient execution of computations across multiple devices and CPUs/GPUs.

Automatic differentiation: TensorFlow provides automatic differentiation, which allows for efficient computation of gradients for use in backpropagation algorithms.

High-level APIs: TensorFlow provides high-level APIs, such as Keras, that allow developers to quickly build and train complex models with minimal code.

Preprocessing and data augmentation: TensorFlow provides a range of tools for preprocessing and data augmentation, including image and text preprocessing, data normalization, and more.

Distributed training: TensorFlow supports distributed training across multiple devices, CPUs, and GPUs, allowing for faster training times and more efficient use of resources.

Model deployment: TensorFlow allows for easy deployment of models to a variety of platforms, including mobile devices and the web.

Visualization tools: TensorFlow provides a range of visualization tools for analyzing model performance, including TensorBoard, which allows for real-time visualization of model training and performance.

4.3 PYCHARM

PyCharm is an integrated development environment (IDE) for Python programming language, developed by JetBrains. PyCharm provides features such as code completion, debugging, code analysis, refactoring, version control integration, and more to help developers write, test, and debug their Python code efficiently. PyCharm is available in two editions: Community Edition (CE) and Professional Edition (PE). The Community Edition is a free, open-source version of the IDE that provides basic functionality for Python development. The Professional Edition is a paid version of the IDE that provides advanced features such as remote development, web development, scientific tools, database tools, and more. PyCharm is available for Windows, macOS, and Linux operating systems. It supports Python versions 2.7, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, and 3.10.

Features:

- Intelligent code completion
- Syntax highlighting

- Code inspection
- Code navigation and search
- Debugging
- Testing
- Version control integration
- Web development support
- Scientific tools support
- Database tools support

Integration with other JetBrains tools

PyCharm's code completion feature can help speed up development by automatically suggesting code based on context and previously written code. It also includes a debugger that allows developers to step through code, set breakpoints, and inspect variables. PyCharm has integration with version control systems like Git, Mercurial, and Subversion. It also supports virtual environments, which allow developers to manage different Python installations and packages in isolated environments. The IDE also has features specifically geared towards web development, such as support for popular web frameworks like Django, Flask, and Pyramid. It includes tools for debugging, testing, and profiling web applications. PyCharm also provides scientific tools for data analysis, visualization, and scientific computing, such as support for NumPy, SciPy, and matplotlib. It also includes tools for working with databases, such as PostgreSQL, MySQL, and Oracle. Overall, PyCharm is a powerful and feature-rich IDE that can greatly increase productivity for Python developers.

Customization:

PyCharm allows developers to customize the IDE to their liking. Users can change the color scheme, fonts, and other settings to make the IDE more comfortable to use. PyCharm also supports plugins, which allow developers to extend the IDE with additional features.

Collaboration:

PyCharm makes it easy for developers to collaborate on projects. It supports integration with popular collaboration tools such as GitHub, Bitbucket, and GitLab. It also includes features for code reviews, task management, and team communication.

Education:

PyCharm provides a learning environment for Python programming language. PyCharm Edu is a free, open-source edition of PyCharm that includes interactive courses and tutorials for learning Python. It provides an easy-to-use interface for beginners and includes features such as code highlighting, autocompletion, and error highlighting.

Support:

PyCharm has an active community of users who provide support through forums and social media. JetBrains also provides comprehensive documentation, tutorials, and training courses for PyCharm. For users who need more personalized support, JetBrains offers a paid support plan that includes email and phone support.

Pricing:

PyCharm Community Edition is free and open-source. PyCharm Professional Edition requires a paid license, but offers a 30-day free trial. JetBrains also offers a subscription-based pricing model that includes access to all JetBrains IDEs and tools.

Integrations:

PyCharm integrates with a wide range of tools and technologies commonly used in Python development. It supports popular Python web frameworks like Flask, Django, Pyramid, and web2py. It also integrates with tools for scientific computing like NumPy, SciPy, and pandas. PyCharm also supports popular front-end technologies such as HTML, CSS, and JavaScript.

Performance:

PyCharm is known for its fast and reliable performance. It uses a combination of static analysis, incremental compilation, and intelligent caching to provide fast code completion and navigation. PyCharm also has a memory profiler that helps identify and optimize memory usage in Python applications.

Ease of Use:

PyCharm provides an intuitive and easy-to-use interface for developers. It has a well-organized menu structure, clear icons, and easy-to-navigate tabs. PyCharm also provides a variety of keyboard shortcuts and customizable keymaps that allow users to work efficiently without constantly switching between the mouse and keyboard.

Community:

PyCharm has a large and active community of developers who contribute to the development of the IDE. The PyCharm Community Edition is open-source, which means that

anyone can contribute to its development. The PyCharm user community is also active in providing support, tips, and tutorials through forums, blogs, and social media.

MY SQL

MySQL is the world's most used open source relational database management system (RDBMS) as of 2008 that run as a server providing multi-user access to a number of databases. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack—LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open-source projects that require a full-featured database management system often use MySQL.For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, Word Press, phpBB, MyBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google(though not for searches), Imagebook Twitter, Flickr, Nokia.com, and YouTube.

Inter images

MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools, or use MySQL "front-ends", desktop software and web applications that create and manage MySQL databases, build database structures, back up data, inspect status, and work with data records. The official set of MySQL front-end tools, MySQL Workbench is actively developed by Oracle, and is freely available for use.

Graphical

The official MySQL Workbench is a free integrated environment developed by MySQL AB that enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL frontend,

MySQL Workbench lets users manage database design & modeling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).MySQL Workbench is available in two editions, the regular free and open source Community Edition which may be downloaded from the MySQL website, and the proprietary Standard Edition which extends and improves the feature set of the Community Edition.

CHAPTER 5

5. PROJECT DESCRIPTION

5.1 PROBLEM DESCRIPTION

The implementation of an automated road crack detection system is imperative due to the limitations and challenges associated with conventional methods of road inspection. Traditional manual inspections are often time-consuming, resource-intensive, and prone to delays in identifying critical issues, leading to safety hazards and increased maintenance costs. An automated system addresses these shortcomings by enabling the early detection of cracks, allowing for timely intervention and preventive measures. This not only enhances overall road safety by minimizing the risk of accidents but also proves cost-efficient by optimizing maintenance expenses. Proactive crack detection helps avoid disruptions to traffic flow, as automated systems can efficiently analyse large stretches of road without causing significant delays. In essence, the need for an automated road crack detection system is grounded in its ability to enhance safety, reduce costs, and streamline maintenance processes, ultimately contributing to the longevity and sustainability of road infrastructure.

The deployment of a road crack detection system faces several challenges that necessitate careful consideration for effective functionality. One primary challenge involves the variability in road surfaces, where different materials and textures can affect the consistency of crack detection. The system must be designed to accommodate and adapt to this diversity to maintain accuracy across a spectrum of road types. Additionally, adverse weather conditions, such as rain, snow, or fog, present a significant hurdle, impacting the visibility of road surfaces and potentially compromising the performance of optical sensors. To ensure the reliability of crack detection under diverse weather conditions, the system must exhibit resilience and robustness. Another critical challenge is associated with the quality of data obtained from sensors or imaging devices. Factors like poor lighting, shadows, or sensor noise can introduce inaccuracies, demanding the implementation of algorithms that can effectively distinguish between genuine cracks and data artifacts. Addressing these challenges is crucial for the successful deployment of a road crack detection system that can operate accurately and consistently in real-world conditions, contributing to the improvement of road safety and infrastructure maintenance.

5.2 DATASETS DESCRIPTION

This dataset contains images of roads with damages. We classify the damages of roads into 4 categories such as good, poor, satisfactory, very poor according to their extent of damage. And can use labels to the images by the name given by the directories or by the prefix of name of the image. And can use these images to classify the roads and also can be used for further assessment.

5.3 MODULES DESCRIPTION

5.3.1 TRAINING AS A DATASETS

Image-based systems have several benefits for monitoring the crack propagation in different structural material. Initially, when these systems were used to measurement of cracks, more attention was paid to features of objects and repeatability. Also, the use of remote sensing techniques allows the measurement of cracks without the need for access to the validated elements, and also provides stable image storage for each observation in any period. These systems are helpful for those who are involved in the design of structures or those who are responsible for maintaining the infrastructure systems when analyzing the relationship between loading and damage locations. In this module, we can input the image with any type and any size.

5.3.2 IMAGE NORMALIZATION

Pre-processing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. These iconic images are of the same kind as the original data captured by the sensor, with an intensity image usually represented by a matrix of image function values (brightnesses). Image pre-processing methods are classified into four categories according to the size of the pixel neighborhood that is used for the calculation of new pixel brightness. The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, translation) are classified among pre-processing methods here since similar techniques are used. The user has to select the required frame image for further processing. In this module convert the RGB image into gray scale image. The colors of images are always gray shades and the variety of changes in atmosphere cause the color feature having low reliability. The formula used for converting the RGB pixel value to its grey scale counterpart is given in Equation.

$$Gray = 0.2989 * R + 0.5870 * G + 0.1140 * B$$

where R, G, B correspond to the color of the pixel, respectively.

Median filtering is a prominent image processing technique employed for noise reduction and edge preservation. In this method, a sliding window traverses the entire image, and for each pixel, the neighboring pixel values within the window are sorted. The central pixel is then assigned the median value from this sorted set. Typically using window sizes like 3x3 or 5x5, median filtering is particularly effective in mitigating salt-and-pepper noise, where isolated bright and dark pixels randomly pepper the image. Unlike mean filtering, which replaces the central pixel with the average of the surrounding values, median filtering is robust against outliers. It excels in preserving edges in images, making it well-suited for applications where maintaining sharp details is crucial.

Define the Window Size: Choose an appropriate window size for the median filter. Common choices are 3x3, 5x5, or 7x7, depending on the level of noise and the desired filtering effect.

Iterate Through Pixels: Start iterating through each pixel in the image.

Select Neighborhood: For each pixel, identify the neighboring pixels within the defined window. If the window extends beyond the image boundaries, it is common to consider only the valid pixels within the image.

Sort Pixel Values: Collect the pixel values within the neighborhood and sort them in ascending order. The central pixel's value will be replaced with the median value of this sorted set.

Calculate Median: Determine the median value from the sorted set of neighboring pixel values. If the number of pixels in the neighborhood is even, take the average of the two middle values.

Replace Pixel Value: Replace the original value of the central pixel with the calculated median value.

Repeat for All Pixels: Continue this process for every pixel in the image, excluding the pixels near the borders where the full neighborhood might not be available.

Complete Filtering: Once all pixels have been processed, the median-filtered image is generated.

5.3.3 DIMENSIONALITY REDUCTION

CNN has significant application in crack detection. The purpose of CNN is to generate a network system with little errors but also yield good result from the testing data set. In this module implement artificial neural network algorithm to classify image as normal or cracked. The neural network itself isn't an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. A CNN is based on a collection of connected units or nodes called artificial neurons, which loosely

model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common CNN implementations, the signal at a connection between artificial neurons are a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold.

Step 1: Randomly initialize the weights and biases.

Step 2: feed the training sample.

Step 3: Propagate the inputs forward; compute the net input and output of each unit in the hidden and output layers.

Step 4: back propagate the error to the hidden layer.

Step 5: update weights and biases to reflect the propagated errors.

Training and learning functions are mathematical procedures used to automatically adjust the network's weights and biases.

Step 6: terminating condition

Based on these steps, model file is generated and used for further purpose.

5.3.4 MODEL CONSTRUCTION

Road damage classification can be evaluated by plotting a confusion metrics. Confusion matrix will help us to find out the relationship between predicted and actual values for each target attributes. When we compare the confusion matrix of balance dataset with imbalanced data using CNN model, we can observe that model is able to identify all the labels. Accuracy Precision refers to the ratio of accurate forecasts to those that were supposed to be right but weren't. The number of right predictions divided by several accurate forecasts and models was supposed to be distinct but was not equal recall. The following formulae may be used to calculate the accuracy, recall, and f-score.

In experimental results, from key features datasets, the sign facts are acquired which are employed to measure the usefulness of the suggested method. Using F-measure, Recall and Precision the performance of the system is being evaluated.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F measure} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Accuracy (ACC) is found as the fraction of total number of perfect predictions to the total number of test data. It can also be represented as 1 – ERR. The finest possible accuracy is 1.0, whereas the very worst is 0.0.

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} \text{ x } 100$$

5.3.5 DAMAGE CLASSFICATION

Road infrastructure is susceptible to the formation of cracks over time, leading to safety hazards, increased maintenance expenditures, and potential disruptions to traffic. The manual inspection of roads for crack detection is time-consuming, resource-intensive, and often results in delayed identification of critical issues. This hinders the ability to proactively address road defects, increasing the risk of accidents and necessitating more extensive and costly repairs. Moreover, traditional methods of road inspection may not be able to keep pace with the rapid expansion of road networks and the growing demand for timely maintenance. In this module, user input the image about road datasets. Road image can be classified with model file. Model classification using CNN algorithm. Finally provide the alert about damages with types and also send complaints about road damage to concern authorities

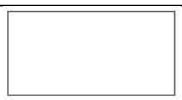
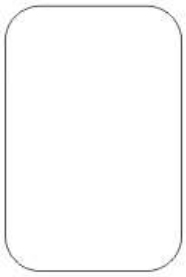
CHAPTER 6

6. DIAGRAMS

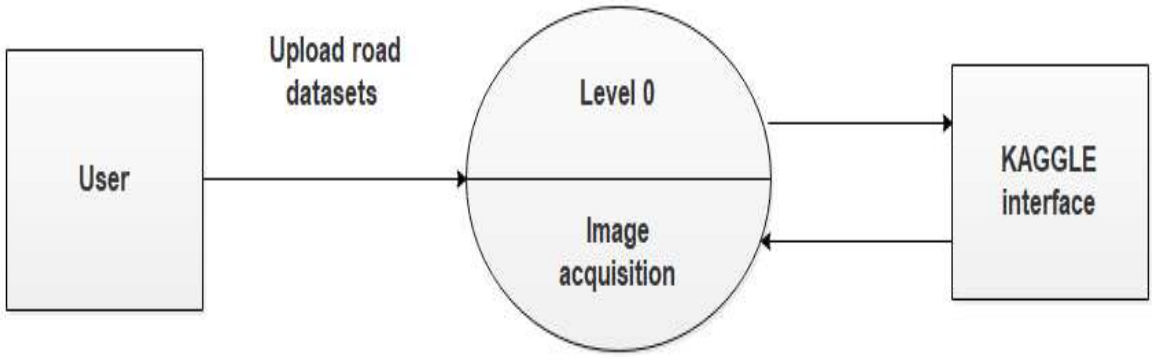
6.1 DATA FLOW DIAGRAM

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

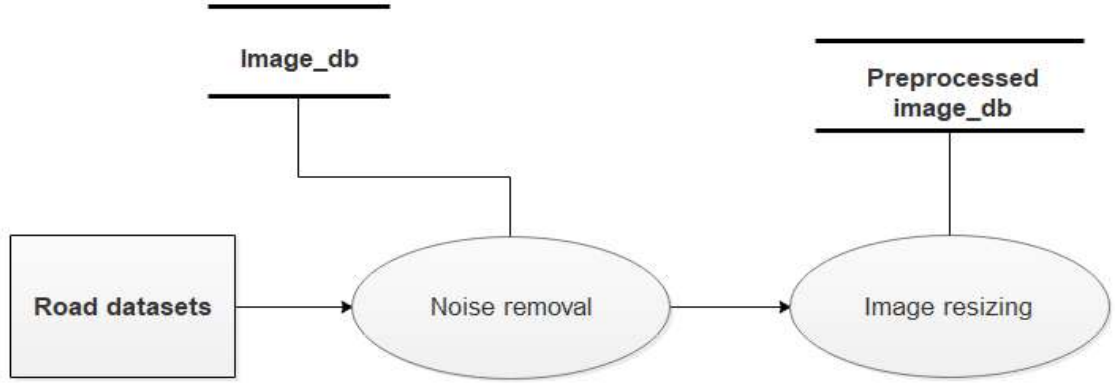
Data flow Symbols:

Symbol	Description
	An **entity**. A source of data or a destination for data.
	A **process** or task that is performed by the system.
	A **data store**, a place where data is held between processes.
	A **data flow**.

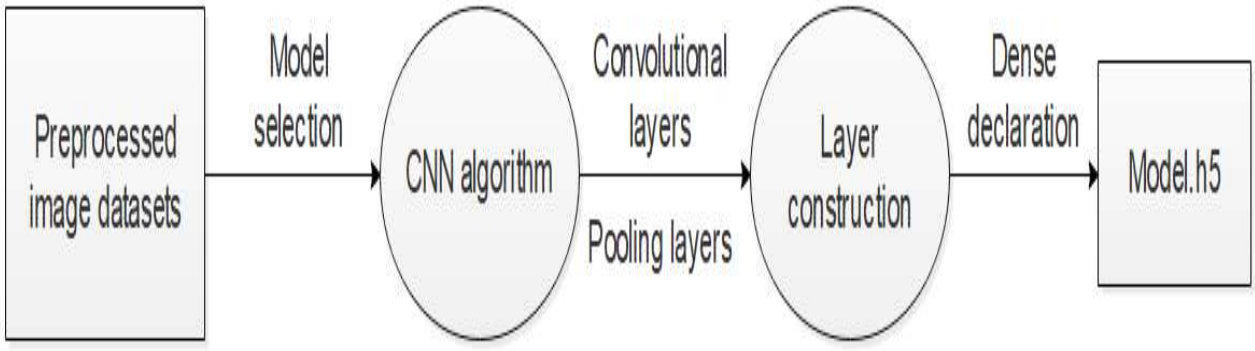
LEVEL 0



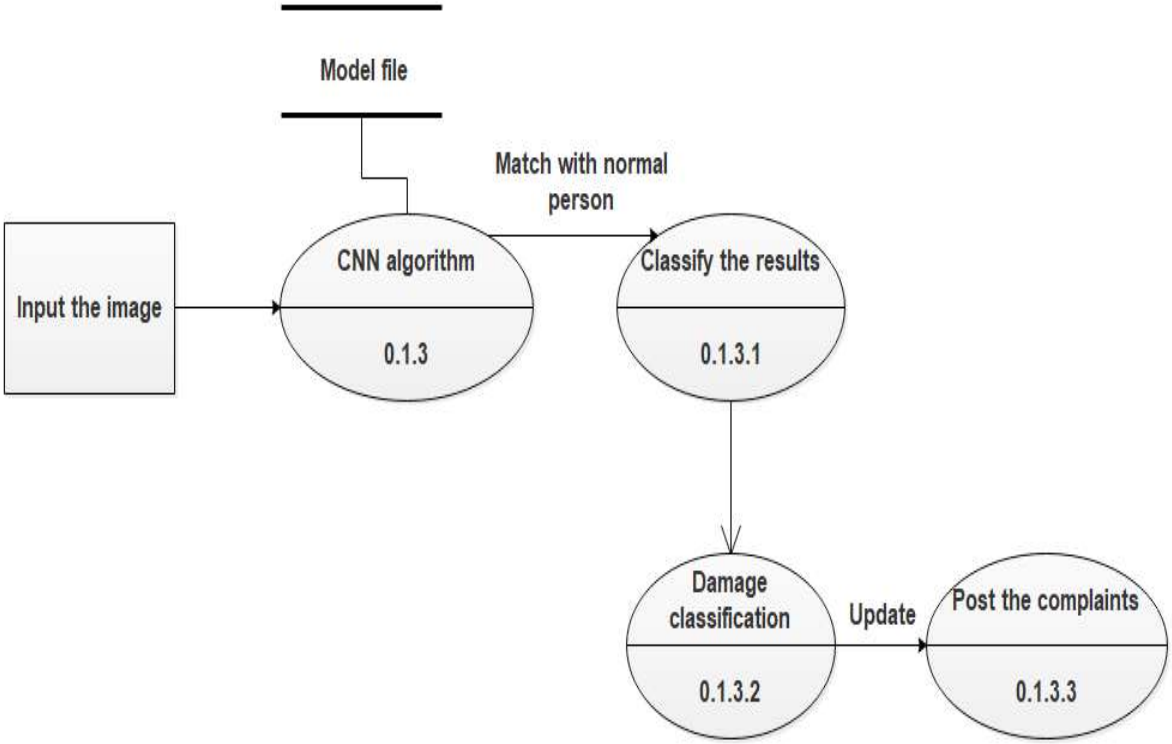
LEVEL 1



LEVEL 2



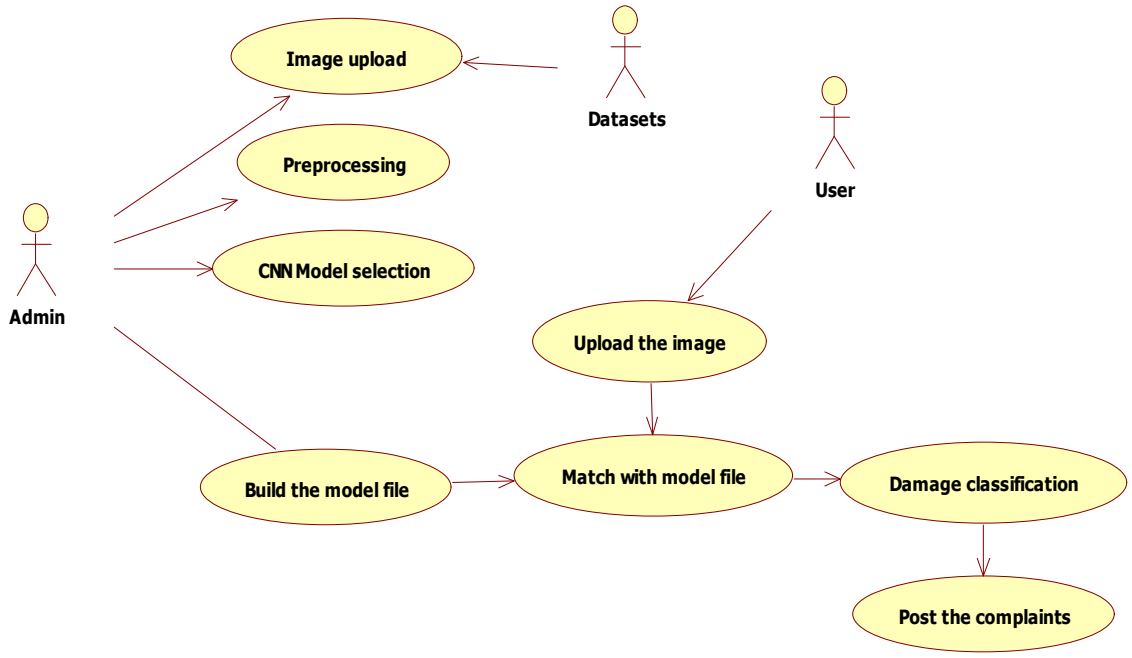
LEVEL 3


Model file
Match with normal person
Input the image
CNN algorithm
0.1.3
Classify the results
0.1.3.1
Damage classification
0.1.3.2
Update
Post the complaints
0.1.3.3

6.2 UML DIAGRAMS

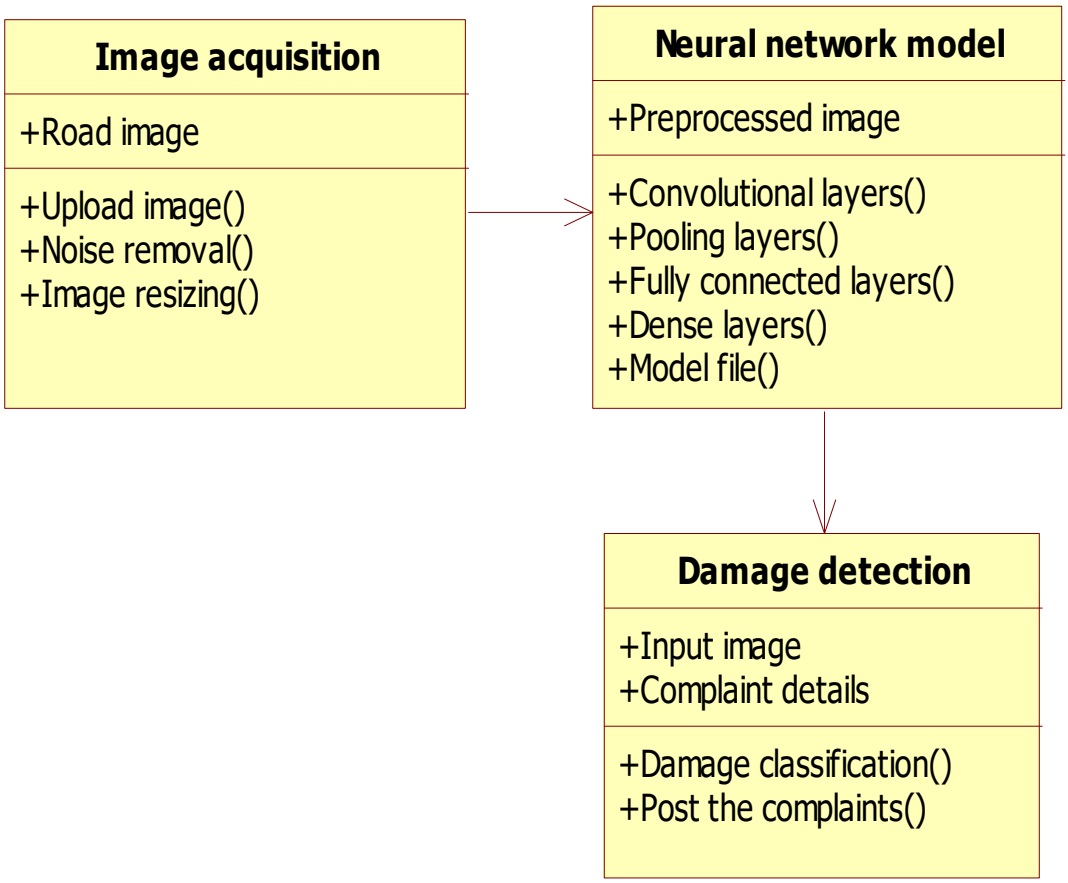
6.2.1 USECASE DIAGRAM

A use case diagram is a type of UML diagram that represents the interactions between an actor (a user or system) and a system under various scenarios. The diagram provides a visual representation of the system's functionalities and the interactions between the actors and the system.



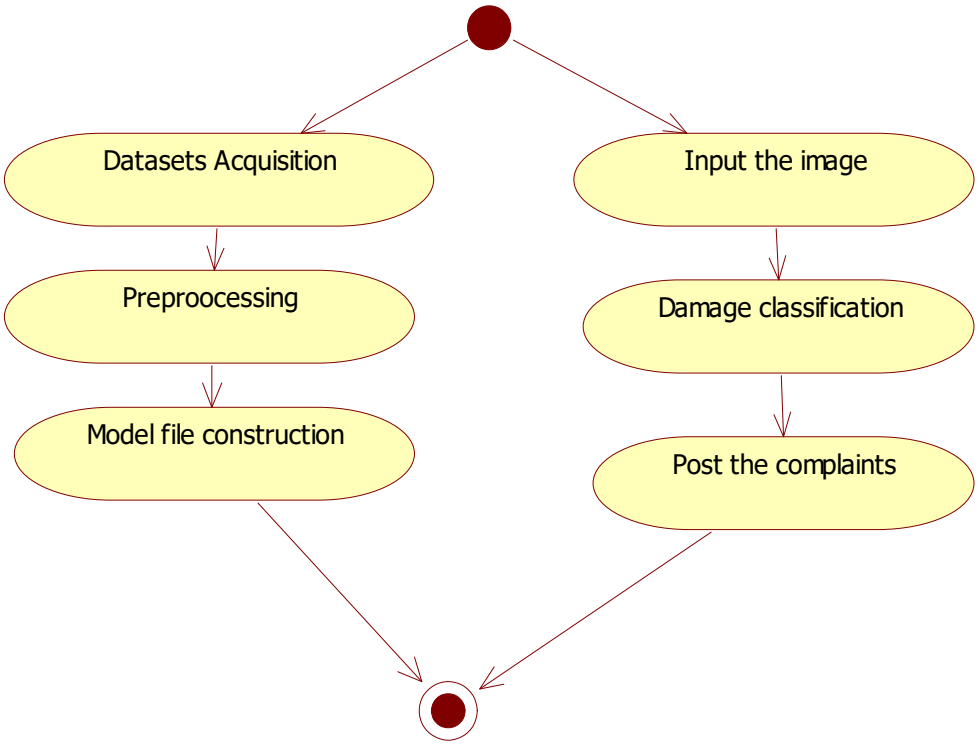
6.2.2 CLASS DIAGRAM

A class diagram is a type of visual representation used in software development to depict the classes, attributes, and methods of a system, as well as the relationships that exist between them. In this type of diagram, each class is represented as a box that includes its name, along with its attributes and methods. The attributes of a class are represented as variables that describe the characteristics of the class, while methods are represented as functions that define the behavior of the class.



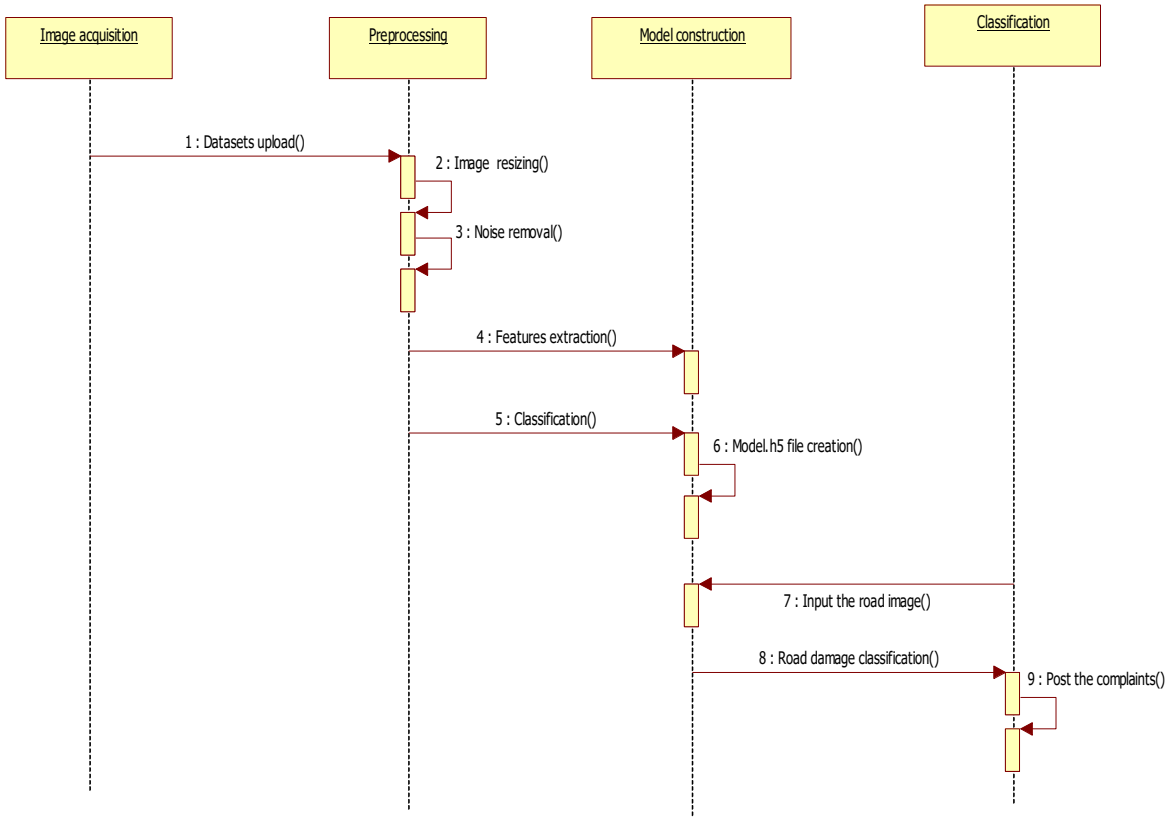
6.2.3 ACTIVITY DIAGRAM

Activity diagrams show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity. They may be used to detail situations where parallel processing may occur in the execution of some activities.



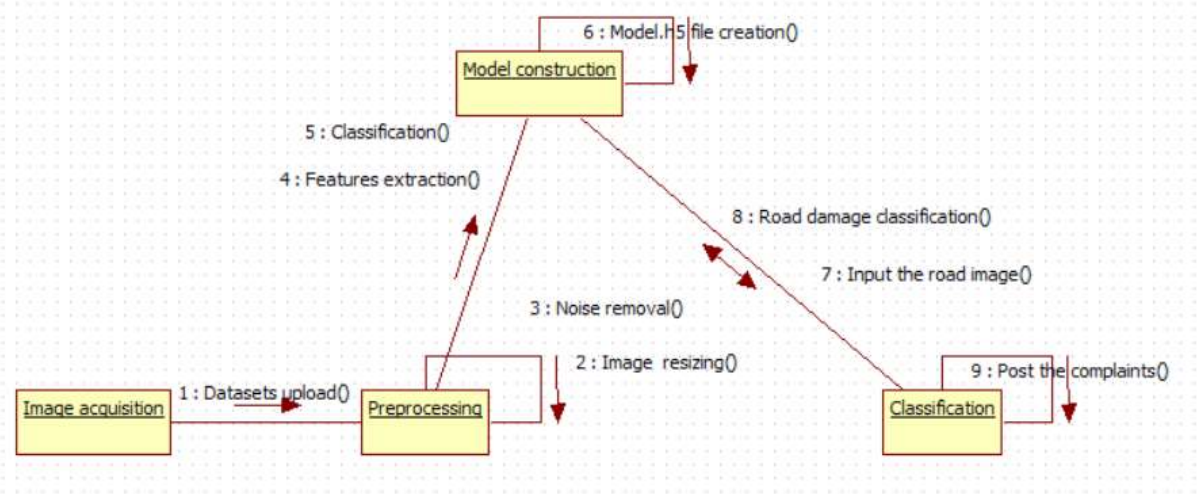
6.2.4 SEQUENCE DIAGRAM

A sequence diagram is a type of UML (Unified Modeling Language) diagram that illustrates the interactions between objects in a system in a sequential order. It shows the sequence of messages exchanged between objects over time, and can be used to represent a variety of scenarios, such as the flow of control between objects, the interactions between different components of a system, and the sequence of events that occur during a particular process.



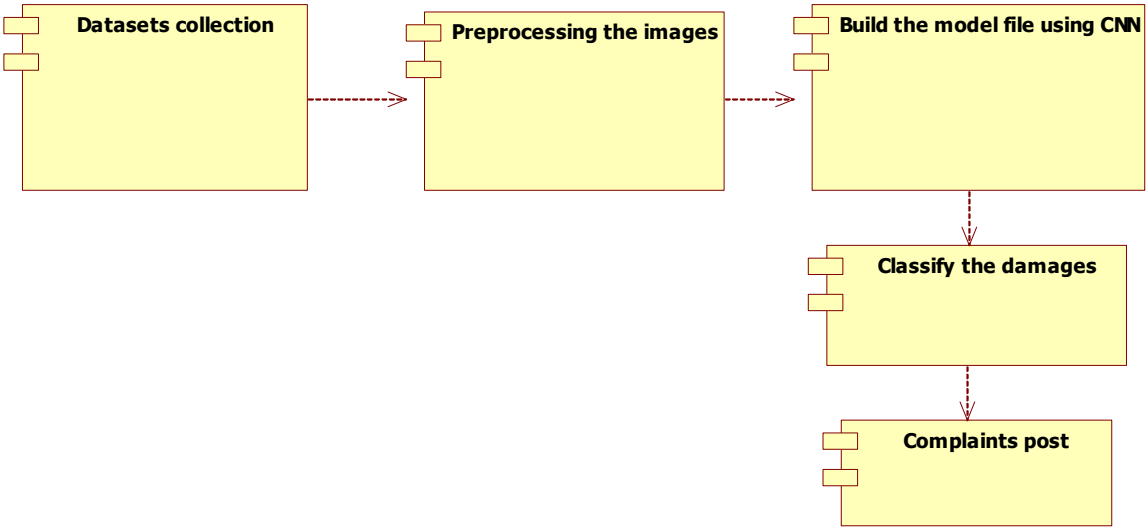
6.2.5 COLLBORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

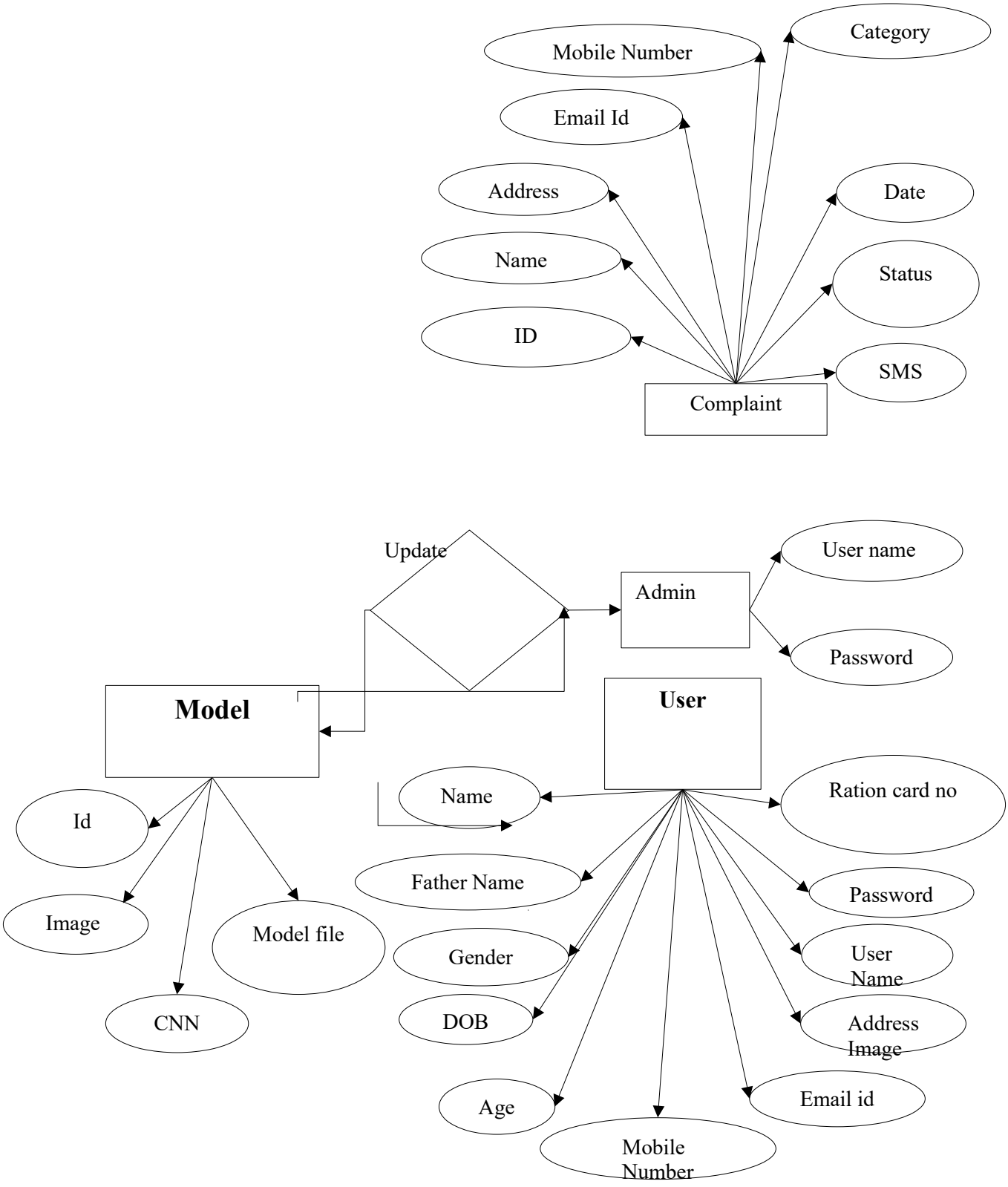


6.2.6 COMPONENT DIAGRAM

In Unified Modeling Language (UML), a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.



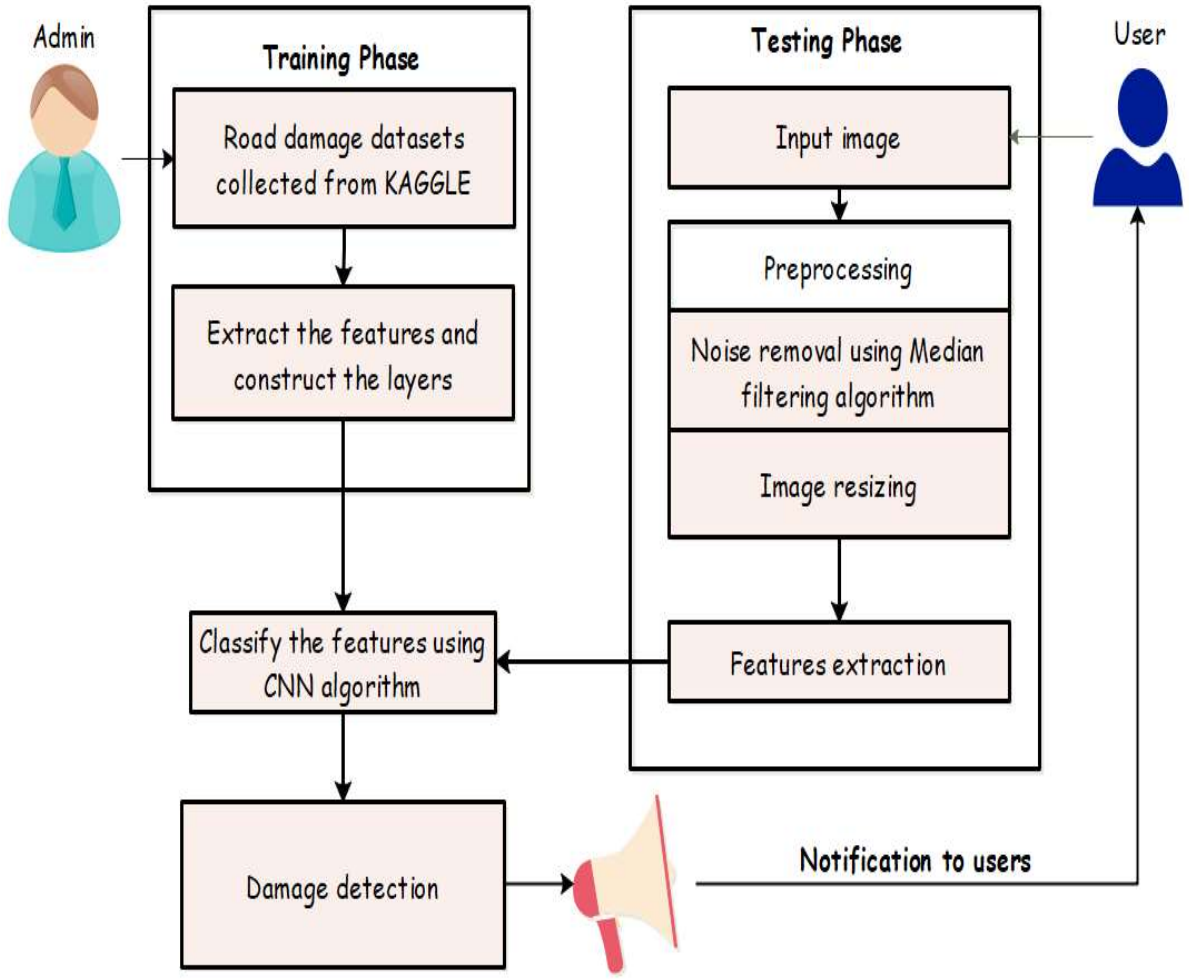
6.2.7 ER DIAGRAM



CHAPTER 7

7. SYSTEM ARCHITECTURE

7.1 OVERALL ARCHTIECTURE



CHAPTER 8

8. SYSTEM TESTING

8.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. Test objectives

8.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up software applications at the company level interact without error.

8.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centred on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.

- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.4 System Test

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

8.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.7 Acceptance Testing

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

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 9

9. SYSTEM IMPLEMENTATION

9.1 CODING

```
from flask import Flask, render_template, flash, request, session


import cv2


app = Flask(__name__)
app.config.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'



@app.route("/")
def homepage():
    return render_template('index.html')



@app.route("/Prediction")
def Prediction():
    return render_template('Prediction.html')



@app.route("/predict", methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':

        file = request.files['file']
        file.save('static/Out/Test.jpg')

        import warnings
        warnings.filterwarnings('ignore')
```

```
        import tensorflow as tf
        classifierLoad = tf.keras.models.load_model('model.h5')

        import numpy as np
        from keras.preprocessing import image

        test_image = image.load_img('static/Out/Test.jpg', target_size=(200, 200))
        # img1 = cv2.imread('static/Out/Test.jpg')
        # test_image = image.img_to_array(test_image)
        test_image = np.expand_dims(test_image, axis=0)
        result = classifierLoad.predict(test_image)
        print(result)
        pre = ''

        if result[0][0] == 1:
            pre = "good ES.AMT:rs0"
        elif result[0][1] == 1:
            pre = "poor ES.AMT:rs5000"
        elif result[0][2] == 1:
            pre = "satisfactory ES.AMT:rs500"
        elif result[0][3] == 1:
            pre = "very_poor ES.AMT:rs50000"

        return render_template('Prediction.html', pre=pre)



if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)



# Part 1 - Building the CNN
```

```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.models import model_from_json
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
batch_size = 32
import seaborn as sns
import numpy
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1/255)

# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
        'Data',  # This is the source directory for training images
        target_size=(200, 200),  # All images will be resized to 200 x 200
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['good','poor','satisfactory','very_poor'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')

test_datagen = ImageDataGenerator(rescale=1/255)

# Flow training images in batches of 128 using train_datagen generator
```

```
test_generator = test_datagen.flow_from_directory(
        'Test',  # This is the source directory for training images
        target_size=(200, 200),  # All images will be resized to 200 x 200
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['good','poor','satisfactory','very_poor'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')
import tensorflow as tf


model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),
    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(128, activation='relu'),
    # 5 output neurons for 4 classes with the softmax activation
```

```
    tf.keras.layers.Dense(4, activation='softmax')
])

model.summary()

from tensorflow.keras.optimizers import RMSprop
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['accuracy'])
total_sample=train_generator.n

n_epochs = 20

history = model.fit_generator(
        train_generator,
        steps_per_epoch=int(total_sample/batch_size),
        epochs=n_epochs,
        verbose=1)
model.save('model.h5')
acc = history.history['accuracy']
loss = history.history['loss']
epochs = range(1, len(acc) + 1)
# Train and validation accuracy
plt.plot(epochs, acc, 'b', label=' accurarcy')
plt.title('accurarcy')
plt.legend()
plt.figure()
# Train and validation loss
plt.plot(epochs, loss, 'b', label=' loss')
plt.title('  loss')
plt.legend()
```

```
plt.show()
'''from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
test_steps_per_epoch = numpy.math.ceil(test_generator.samples / test_generator.batch_size)

predictions = model.predict_generator(test_generator, steps=test_steps_per_epoch)
# Get most likely class
predicted_classes = numpy.argmax(predictions, axis=1)

true_classes = test_generator.classes
class_labels = list(test_generator.class_indices.keys())
print('Classification Report')
report = classification_report(true_classes, predicted_classes, target_names=class_labels)
print(report)



print('confusion matrix')
confusion_matrix= confusion_matrix(true_classes, predicted_classes)
print(confusion_matrix)
sns.heatmap(confusion_matrix, annot = True)
plt.show()"
```

APP.PY

```
from flask import Flask, render_template, request, session, flash

import mysql.connector

app = Flask(__name__)
app.config['SECRET_KEY'] = 'aaa'



@app.route('/')
```

```
def home():
    return render_template('index.html')


@app.route('/AdminLogin')
def AdminLogin():
    return render_template('AdminLogin.html')


@app.route('/OfficerLogin')
def OfficerLogin():
    return render_template('OfficerLogin.html')


@app.route('/UserLogin')
def UserLogin():
    return render_template('UserLogin.html')


@app.route('/NewUser')
def NewUser():
    return render_template('NewUser.html')


@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    error = None
    if request.method == 'POST':
        if request.form['uname'] == 'admin' and request.form['password'] == 'admin':


            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
```

```
            cur = conn.cursor()
            cur.execute("SELECT * FROM regtb ")
            data = cur.fetchall()
            flash("you are successfully Login")
            return render_template('AdminHome.html', data=data)

        else:
            flash("UserName or Password Incorrect!")
            return render_template('AdminLogin.html')



@app.route("/AdminHome")
def AdminHome():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM regtb  ")
    data = cur.fetchall()
    return render_template('AdminHome.html', data=data)



@app.route("/NewOfficer")
def NewOfficer():
    return render_template('NewOfficer.html')



@app.route("/newofficer", methods=['GET', 'POST'])
def newofficer():
    if request.method == 'POST':
        name = request.form['name']
        mobile = request.form['mobile']
        email = request.form['email']
        address = request.form['address']
```

```
        depart = request.form['depart']
        username = request.form['uname']
        password = request.form['password']

        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
        cursor = conn.cursor()
        cursor.execute(
            "insert into officertb values('" + name + "','" + mobile + "','" + email + "','" + address + "','"
+ depart + "','" + username + "','" + password + "')")
        conn.commit()
        conn.close()
        flash("Record Saved!")

    return render_template('NewOfficer.html')


@app.route("/AOfficerInfo")
def AOfficerInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM officertb ")
    data = cur.fetchall()
    return render_template('AOfficerInfo.html', data=data)


@app.route("/AComplaintInfo")
def AComplaintInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM complainttb  ")
    data = cur.fetchall()
```

```
    return render_template('AComplaintInfo.html', data=data)



@app.route("/newuser", methods=['GET', 'POST'])
def newuser():
    if request.method == 'POST':
        name = request.form['name']
        mobile = request.form['mobile']
        email = request.form['email']
        address = request.form['address']
        username = request.form['uname']
        password = request.form['password']


        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
        cursor = conn.cursor()
        cursor.execute(
            "insert into regtb values('" + name + "','" + mobile + "','" + email + "','" + address + "','" +
username + "','" + password + "')")
        conn.commit()
        conn.close()
        flash("Record Saved!")


    return render_template('UserLogin.html')



@app.route("/userlogin", methods=['GET', 'POST'])
def userlogin():
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['uname'] = request.form['uname']
```

```
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from regtb where username='" + username + "' and password='" +
password + "'")
        data = cursor.fetchone()
        if data is None:
            flash('Username or Password is wrong')
            return render_template('UserLogin.html', data=data)


        else:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM regtb where username='" + username + "' and password='"
+ password + "'")
            data = cur.fetchall()
            flash("you are successfully logged in")
            return render_template('UserHome.html', data=data)



@app.route("/NewComplaint")
def NewComplaint():
    return render_template('NewComplaint.html', uname=session['uname'])



@app.route("/predict", methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':


        import random
        file = request.files['file']
```

```
fnew = random.randint(1111, 9999)
savename = str(fnew) + ".png"
file.save("static/upload/" + savename)

session['savename'] = savename

import warnings
warnings.filterwarnings('ignore')

import tensorflow as tf
classifierLoad = tf.keras.models.load_model('model.h5')

import numpy as np
from keras.preprocessing import image

test_image = image.load_img("static/upload/" + savename, target_size=(200, 200))
# img1 = cv2.imread('static/Out/Test.jpg')
# test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis=0)
result = classifierLoad.predict(test_image)
print(result)
pre = ''

if result[0][0] == 1:
    pre = "good "
elif result[0][1] == 1:
    pre = "poor"
elif result[0][2] == 1:
    pre = "satisfactory "
elif result[0][3] == 1:
    pre = "very_poor"
```

```
        session['pre'] =pre


        return render_template('NewComplaint1.html', pre=pre, uname=session['uname'])



@app.route("/newcomplaint", methods=['GET', 'POST'])
def newcomplaint():
    if request.method == 'POST':
        uname = session['uname']
        depart = request.form['depart']
        info = request.form['info']


        savename = session['savename']


        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
        cursor = conn.cursor()
        cursor.execute("SELECT  *  FROM regtb where  username='" + uname + "'")
        data = cursor.fetchone()


        if data:
            mobile = data[2]


        else:
            return 'Incorrect username / password !'



        if session['pre'] =='poor' or  session['pre'] =='very_poor':
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
            cursor = conn.cursor()
            cursor.execute(
```

```
            "insert into complainttb values('','" + uname + "','" + mobile + "','" + depart + "','" + info + "','" + savename + "','','waiting','','')")
            conn.commit()
            conn.close()

            conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM complainttb where username='" + uname + "'  ")
            data = cur.fetchall()
            flash('Complaint Post Successfully!')
            return render_template('UComplaintInfo.html', data=data)
        else:

            flash('Complaint Not Post Result!' +  session['pre'])

            return render_template('NewComplaint1.html', pre=session['pre'], uname=session['uname'])



@app.route("/UComplaintInfo")
def UComplaintInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM complainttb where username='" + session['uname'] + "' and Status='waiting' ")
    data = cur.fetchall()
    return render_template('UComplaintInfo.html', data=data)
```

```
@app.route("/UActionInfo")
def UActionInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM complainttb where username='" + session['uname'] + "' and
Status !='waiting' ")
    data = cur.fetchall()
    return render_template('UActionInfo.html', data=data)



@app.route("/officerlogin", methods=['GET', 'POST'])
def officerlogin():
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['oname'] = request.form['uname']


        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from officertb where username='" + username + "' and password='"
+ password + "'")
        data = cursor.fetchone()
        if data is None:
            flash('Username or Password is wrong')
            return render_template('OfficerLogin.html', data=data)


        else:
            session['depart'] = data[5]
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
```

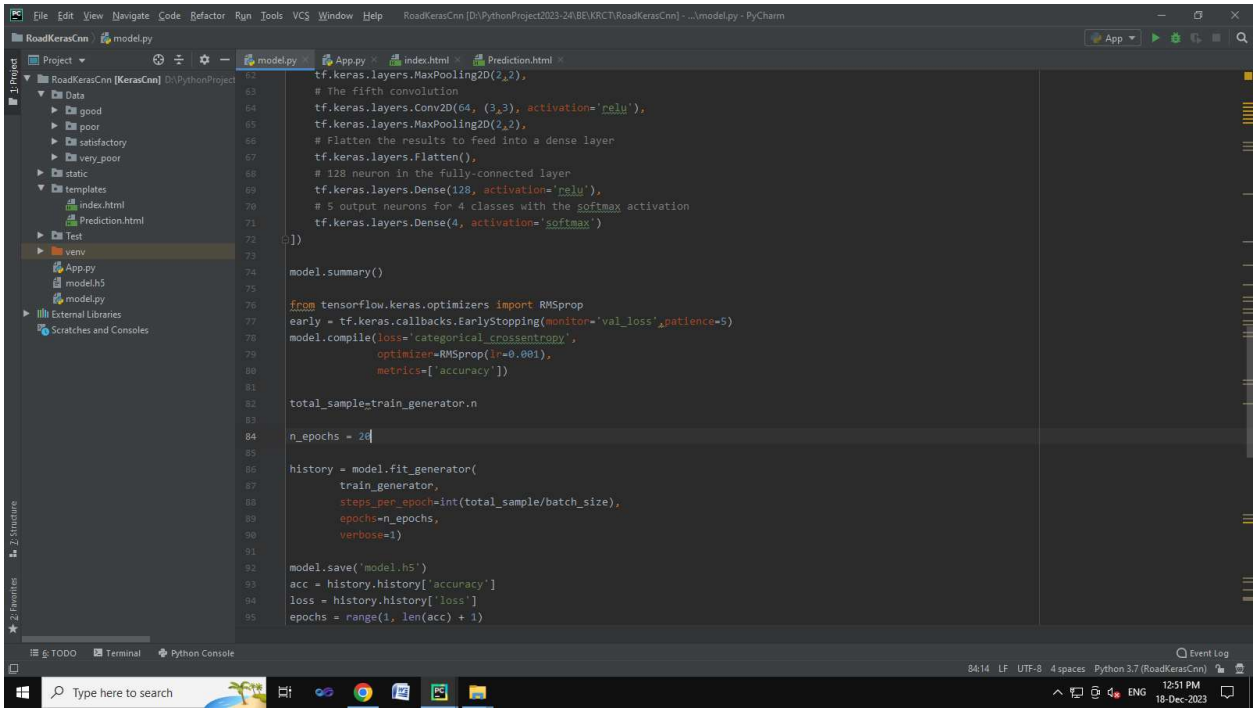
```
        cur = conn.cursor()
        cur.execute("SELECT * FROM officertb where username='" + username + "' and
password='" + password + "'")
        data = cur.fetchall()
        flash("you are successfully logged in")
        return render_template('OfficerHome.html', data=data)


@app.route("/OfficerHome")
def OfficerHome():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM officertb where username='" + session['oname'] + "' ")
    data = cur.fetchall()
    return render_template('OActionInfo.html', data=data)


@app.route("/OActionInfo")
def OActionInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM complainttb where Department='" + session['depart'] + "' and
Status ='completed' ")
    data = cur.fetchall()
    return render_template('OActionInfo.html', data=data)


@app.route("/OComplaintInfo")
def OComplaintInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM complainttb where Department='" + session['depart'] + "' and
Status !='completed' ")
```

```
    data = cur.fetchall()
    return render_template('OComplaintInfo.html', data=data)


@app.route("/action")
def action():
    id = request.args.get('id')
    session["cid"] = id

    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cursor = conn.cursor()
    cursor.execute("SELECT  *  FROM complainttb where  id='" + id + "'")
    data = cursor.fetchone()

    if data:
        mobile = data[2]

    else:
        return 'Incorrect username / password !'

    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1roadpydb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM complainttb where id='" + id + "' ")
    data = cur.fetchall()
    return render_template('Action.html', data=data)


@app.route("/actioninfo", methods=['GET', 'POST'])
def actioninfo():
    if request.method == 'POST':
        act = request.form['act']
        ainfo = request.form['ainfo']
```

```
    oname = session['oname']


    import random
    file = request.files['file']
    fnew = random.randint(1111, 9999)
    savename = str(fnew) + ".png"
    file.save("static/upload/" + savename)


    id = session["cid"]


    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
    cursor = conn.cursor()
    cursor.execute("SELECT  *  FROM complainttb where  id='" + id + "'")
    data = cursor.fetchone()


    if data:
        mobile = data[2]
    else:
        return 'Incorrect username / password !'
    msg = "Your Complaint Action Info" + ainfo
    sendmsg(mobile, msg)


    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1roadpydb')
    cursor = conn.cursor()
    cursor.execute(
        "update   complainttb set Action='" + ainfo + "',Status='" + act + "' , OfficerName='" +
oname + "',Cimage='" + savename + "' where id='" + id + "'")
    conn.commit()
    conn.close()


    flash("Action Info Update successfully")
```

```
        return render_template('OActionInfo.html')



def sendmsg(targetno, message):
    import requests
    requests.post(
        "http://sms.creativepoint.in/api/push.json?
apikey=6555c521622c1&route=transsms&sender=FSSMSS&mobileno=" + targetno +
"&text=Dear customer your msg is " + message + "  Sent By FSMSG FSSMSS")



if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```

9.2 SCREENSHOTS



L:\Python2023\PG\KKC\RoadKerasCnn\venv\Scripts\python.exe D:/PythonProject2023-24/BE/KRCT/RoadKerasCnn/model.py

2023-12-18 12:51:43.549628: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found

2023-12-18 12:51:43.550132: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.

Found 2074 images belonging to 4 classes.

Found 1003 images belonging to 4 classes.

2023-12-18 12:52:10.886894: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2023-12-18 12:52:10.887607: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2023-12-18 12:52:10.911159: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-9BF8NUN

2023-12-18 12:52:10.912126: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-9BF8NUN

2023-12-18 12:52:10.947401: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

_____

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 16)	448
max_pooling2d (MaxPooling2D)	(None, 99, 99, 16)	0
conv2d_1 (Conv2D)	(None, 97, 97, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 32)	0
conv2d_2 (Conv2D)	(None, 46, 46, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 64)	0
conv2d_3 (Conv2D)	(None, 21, 21, 64)	36928
max_pooling2d_3 (MaxPooling	(None, 10, 10, 64)	0

2D)

conv2d_4 (Conv2D) (None, 8, 8, 64) 36928

max_pooling2d_4 (MaxPooling (None, 4, 4, 64) 0

2D)

flatten (Flatten) (None, 1024) 0

dense (Dense) (None, 128) 131200

dense_1 (Dense) (None, 4) 516

===

Total params: 229,156

Trainable params: 229,156

Non-trainable params: 0

_____

Epoch 1/20

WARNING:tensorflow:AutoGraph could not transform <function Model.make_train_function.<locals>.train_function at 0x000001B012EA1E58> and will run it as-is.

Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output.

Cause: 'arguments' object has no attribute 'posonlyargs'

To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert

2023-12-18 12:52:16.309664: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2023-12-18 12:52:17.667029: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 38539264 exceeds 10% of free system memory.

2023-12-18 12:52:18.258781: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 38539264 exceeds 10% of free system memory.

2023-12-18 12:52:18.455965: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 40144896 exceeds 10% of free system memory.

2023-12-18 12:52:18.456174: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

64/64 [==============================] - 100s 2s/step - loss: 0.8456 - accuracy: 0.6660

Epoch 2/20

64/64 [==============================] - 62s 973ms/step - loss: 0.4103 - accuracy: 0.8546

Epoch 3/20

64/64 [==============================] - 56s 853ms/step - loss: 0.3002 - accuracy: 0.8923

Epoch 4/20

64/64 [==============================] - 54s 845ms/step - loss: 0.2457 - accuracy: 0.9246

Epoch 5/20

64/64 [==============================] - 57s 892ms/step - loss: 0.2114 - accuracy: 0.9280

Epoch 6/20

64/64 [==============================] - 66s 1s/step - loss: 0.2032 - accuracy: 0.9344

Epoch 7/20

64/64 [==============================] - 55s 860ms/step - loss: 0.1628 - accuracy: 0.9486

Epoch 8/20

64/64 [==============================] - 65s 1s/step - loss: 0.1479 - accuracy: 0.9432

Epoch 9/20

64/64 [==============================] - 111s 2s/step - loss: 0.1611 - accuracy: 0.9535

Epoch 10/20

64/64 [==============================] - 70s 1s/step - loss: 0.1114 - accuracy: 0.9643

Epoch 11/20

64/64 [==============================] - 68s 1s/step - loss: 0.1131 - accuracy: 0.9638

Epoch 12/20

64/64 [==============================] - 68s 1s/step - loss: 0.0969 - accuracy: 0.9701

Epoch 13/20

64/64 [==============================] - 65s 997ms/step - loss: 0.0820 - accuracy: 0.9750

Epoch 14/20

64/64 [==============================] - 88s 1s/step - loss: 0.0730 - accuracy: 0.9765

Epoch 15/20

64/64 [==============================] - 93s 1s/step - loss: 0.0570 - accuracy: 0.9780

Epoch 16/20

64/64 [==============================] - 92s 1s/step - loss: 0.0590 - accuracy: 0.9819

Epoch 17/20

64/64 [==============================] - 84s 1s/step - loss: 0.0460 - accuracy: 0.9838

Epoch 18/20

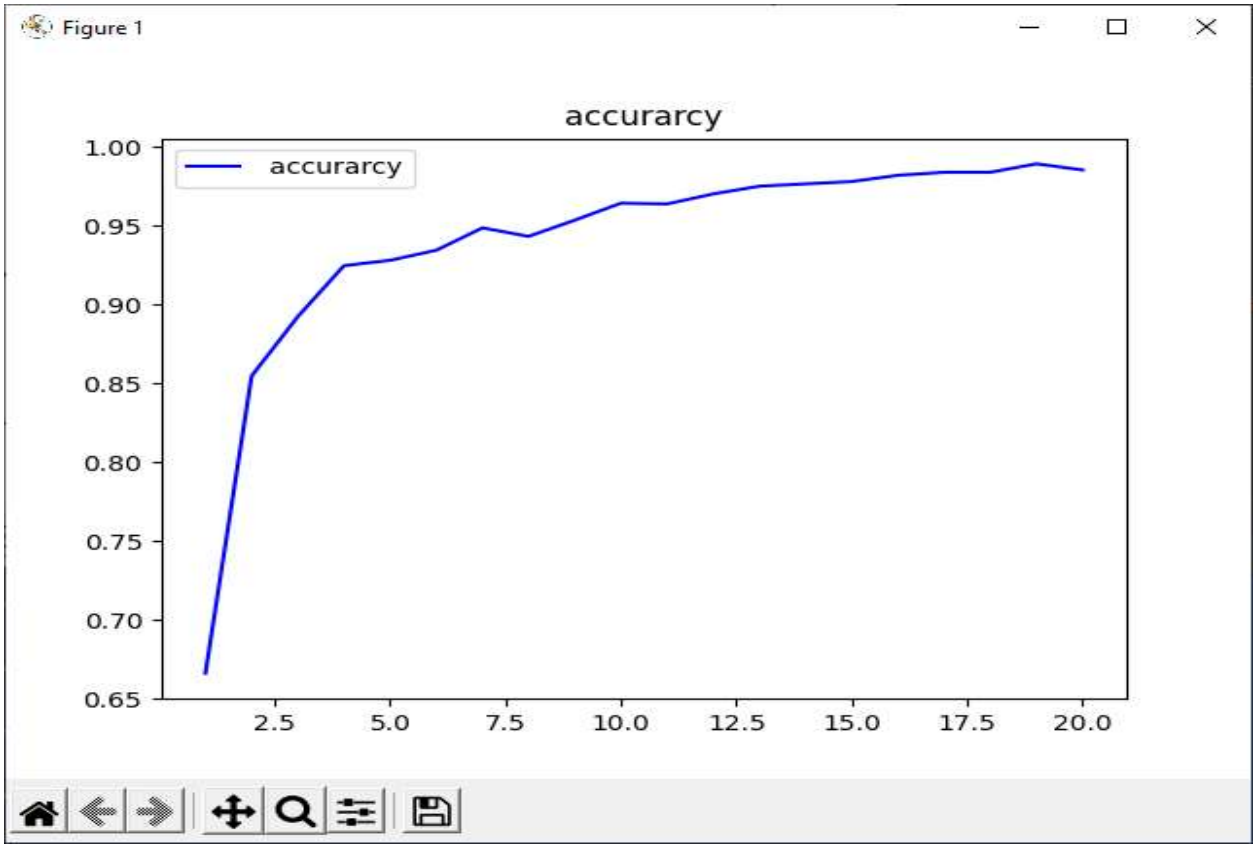
64/64 [==============================] - 58s 899ms/step - loss: 0.0474 - accuracy: 0.9838

Epoch 19/20

64/64 [==============================] - 50s 782ms/step - loss: 0.0406 - accuracy: 0.9892

Epoch 20/20

64/64 [==============================] - 50s 775ms/step - loss: 0.0493 - accuracy: 0.9853



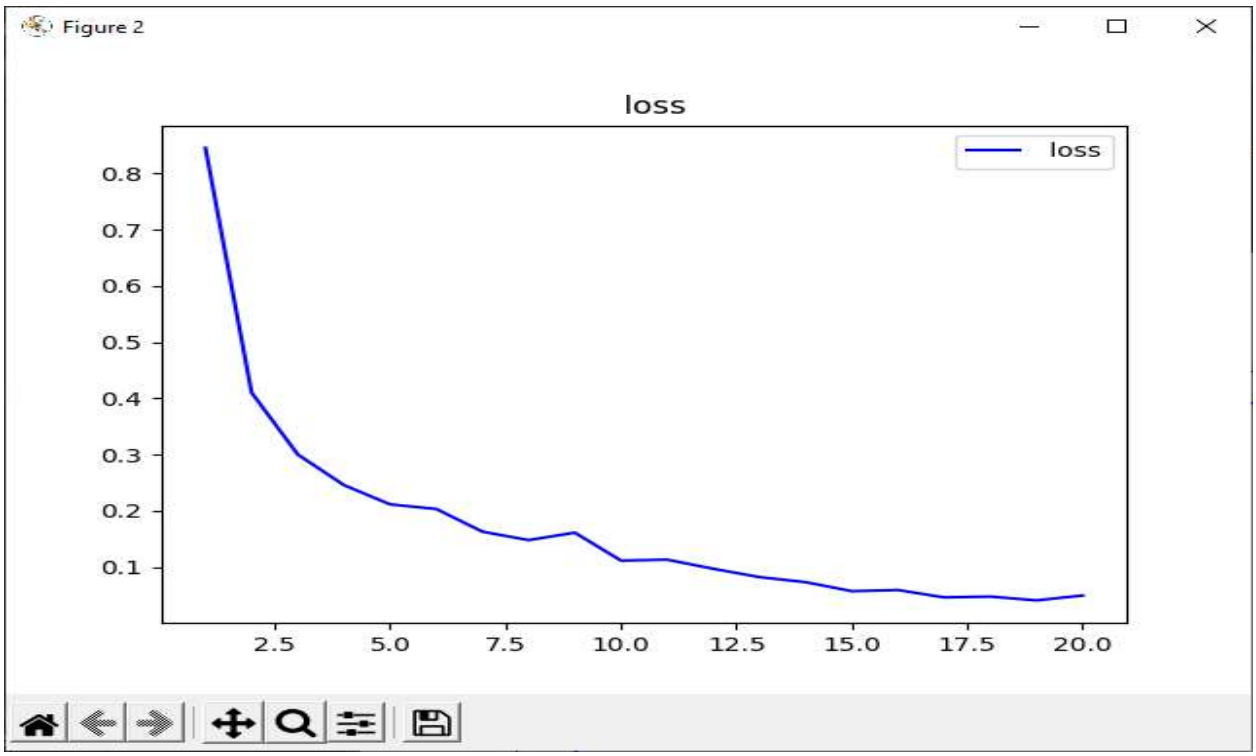
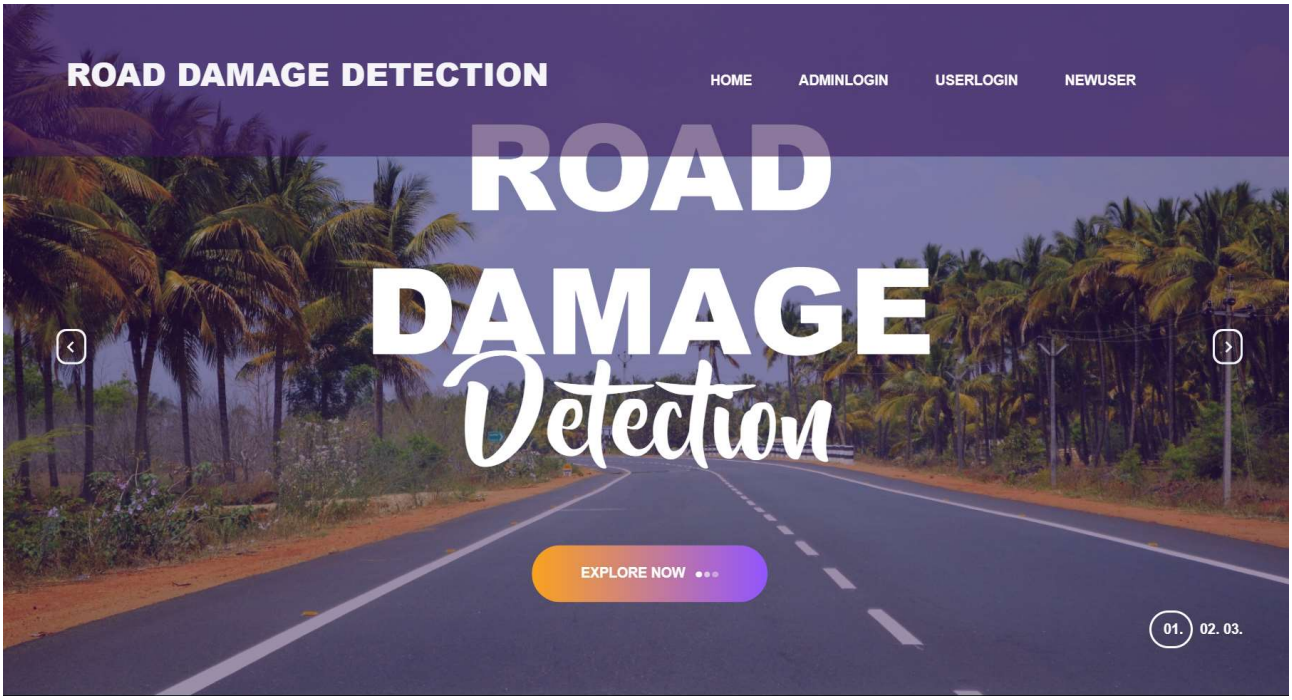
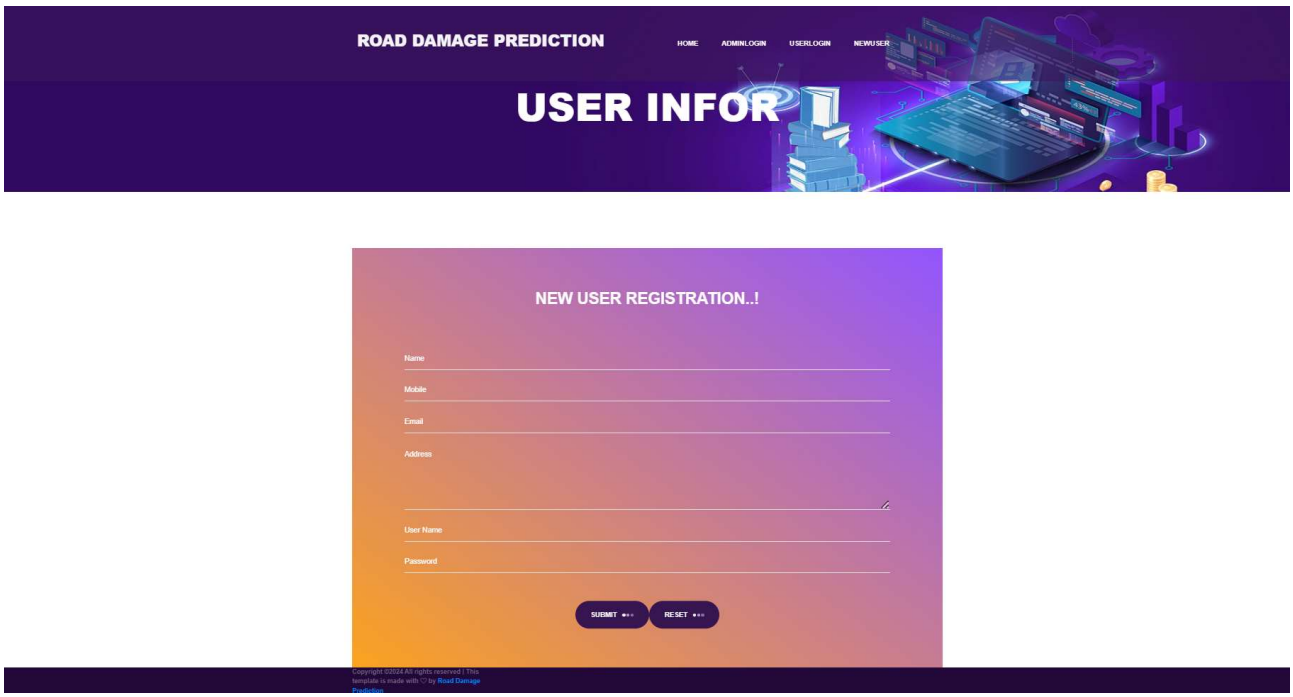
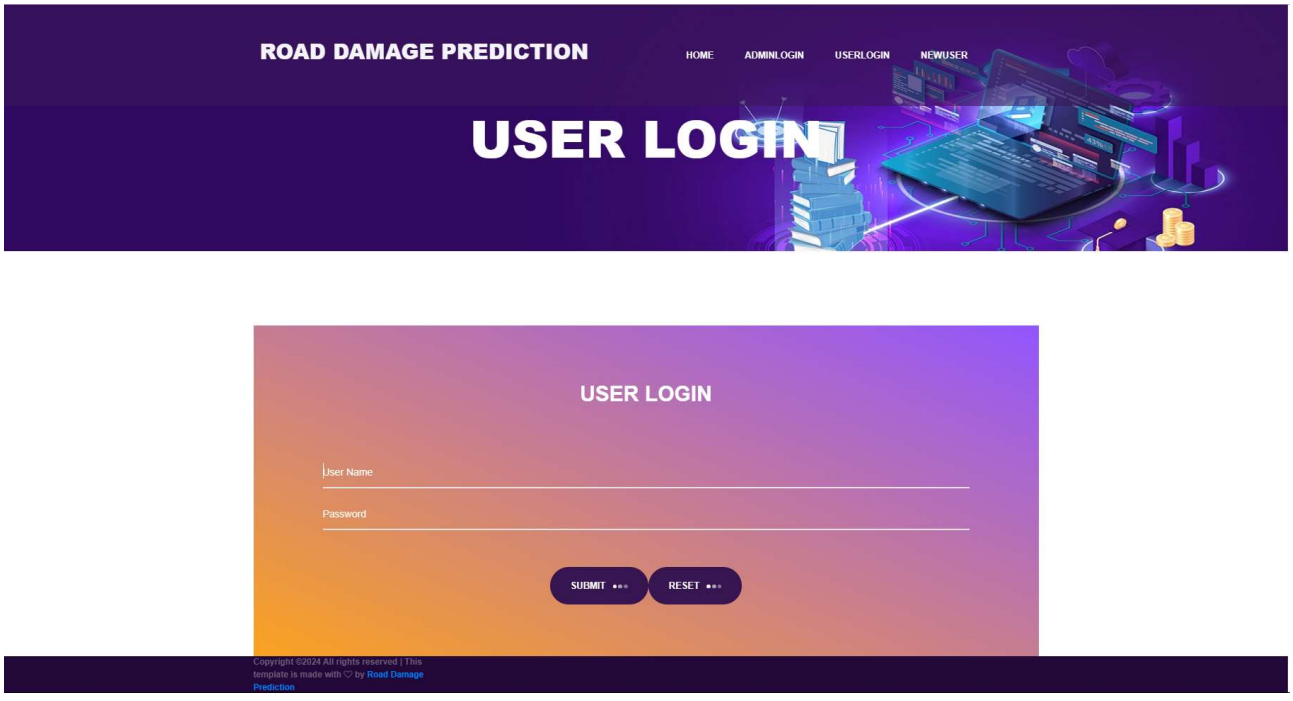
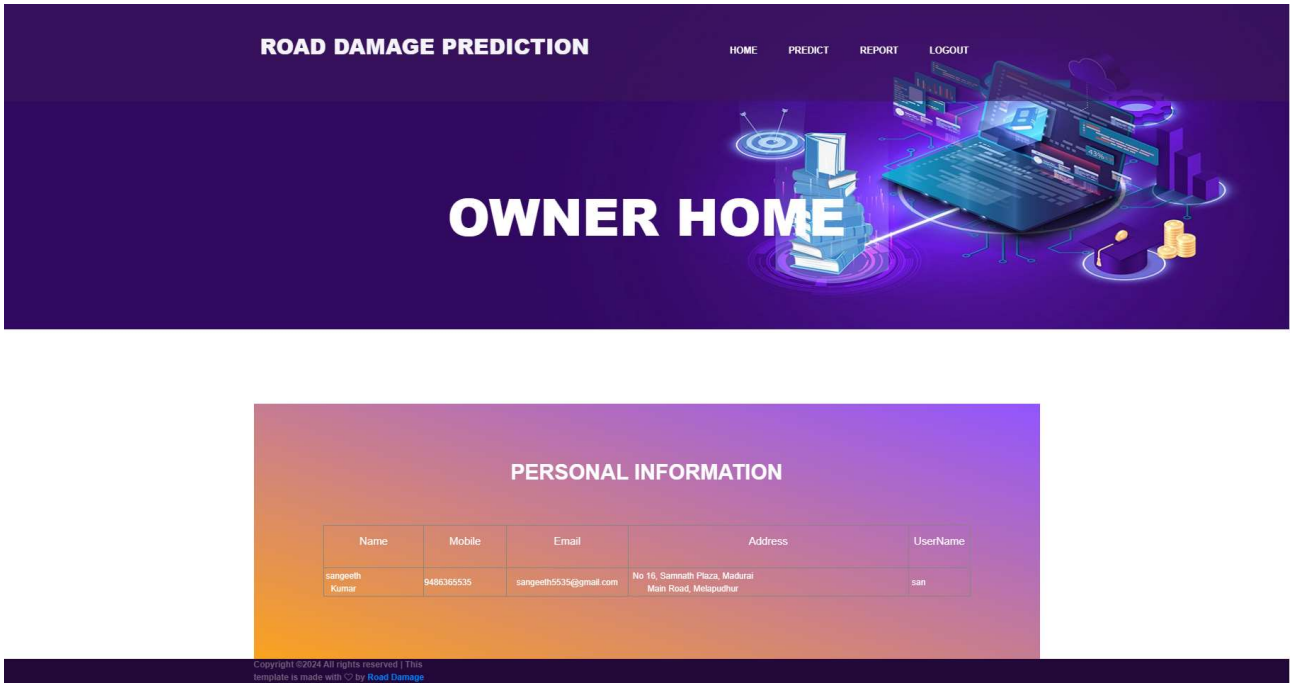


Figure 2
loss
loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

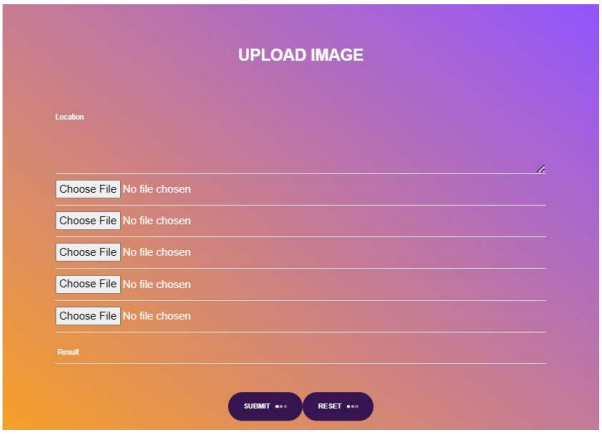

ROAD DAMAGE DETECTION
HOME
ADMINLOGIN
USERLOGIN
NEWUSER
ROAD
DAMAGE
Detection
EXPLORE NOW
01. 02. 03.

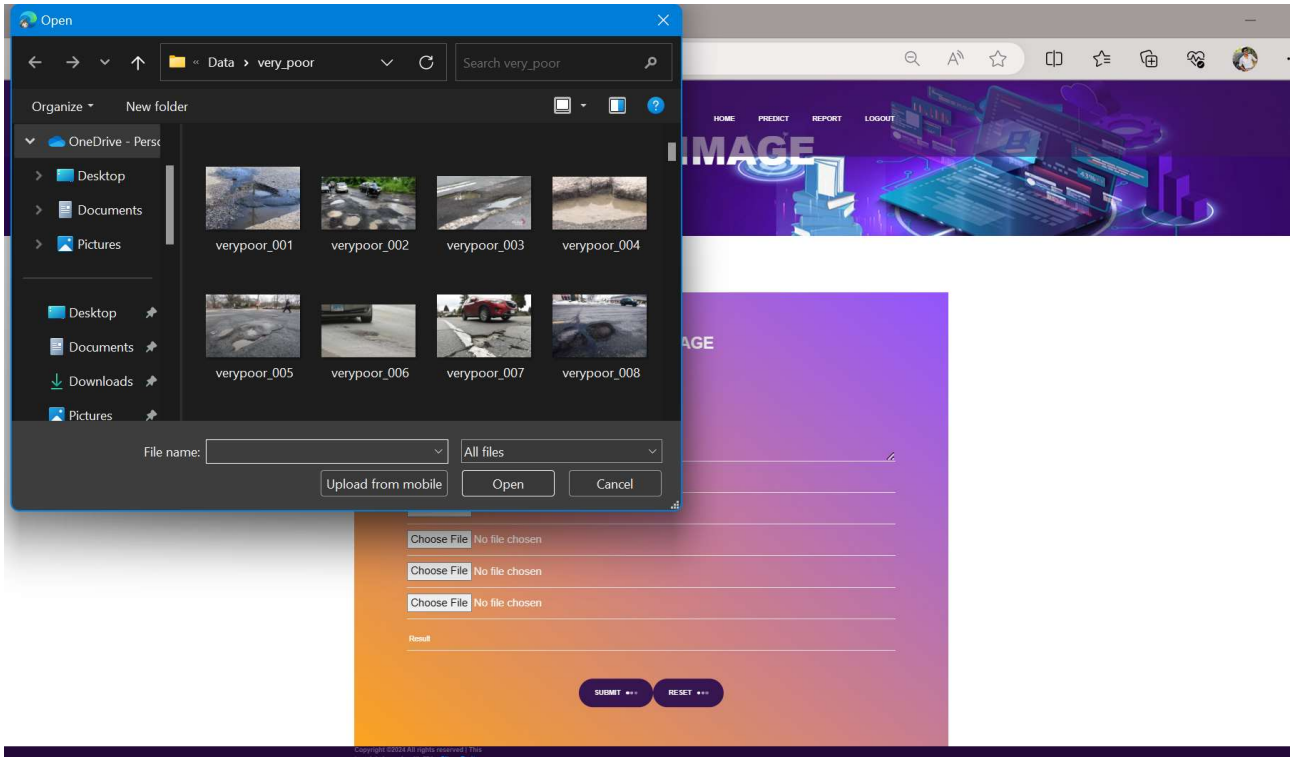

ROAD DAMAGE PREDICTION
HOME
ADMINLOGIN
USERLOGIN
NEWUSER
USER INFOR
NEW USER REGISTRATION..!
Name
Mobile
Email
Address
User Name
Password
SUBMIT
RESET
Copyright ©2024 All rights reserved | This template is made with ♡ by Road Damage Prediction

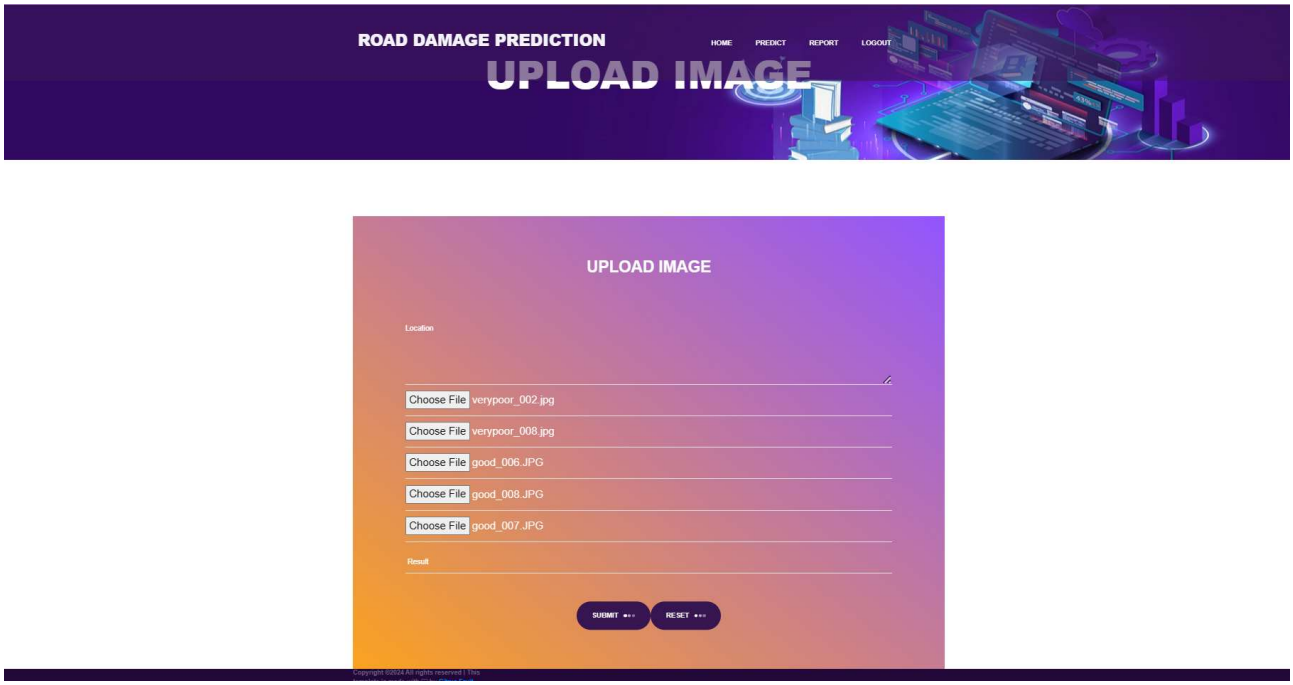

ROAD DAMAGE PREDICTION
HOME
ADMINLOGIN
USERLOGIN
NEWUSER
USER LOGIN
USER LOGIN
User Name
Password
SUBMIT
RESET
Copyright ©2024 All rights reserved | This template is made with ♡ by Road Damage Prediction

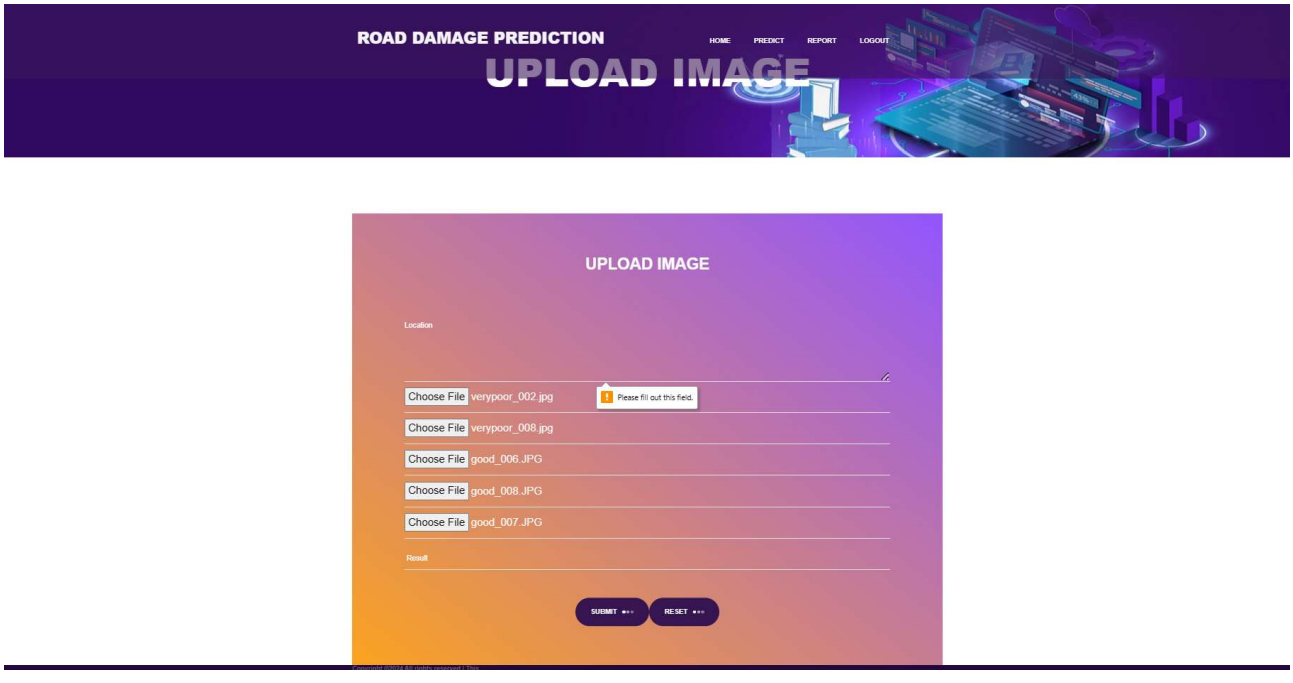

ROAD DAMAGE PREDICTION
HOME
PREDICT
REPORT
LOGOUT
OWNER HOME
PERSONAL INFORMATION
Name
Mobile
Email
Address
UserName
sangeeth Kumar
9486365535
sangeeth5535@gmail.com
No 16, Samnath Plaza, Madurai Main Road, Melapudhur
san
Copyright ©2024 All rights reserved | This template is made with ♡ by Road Damage

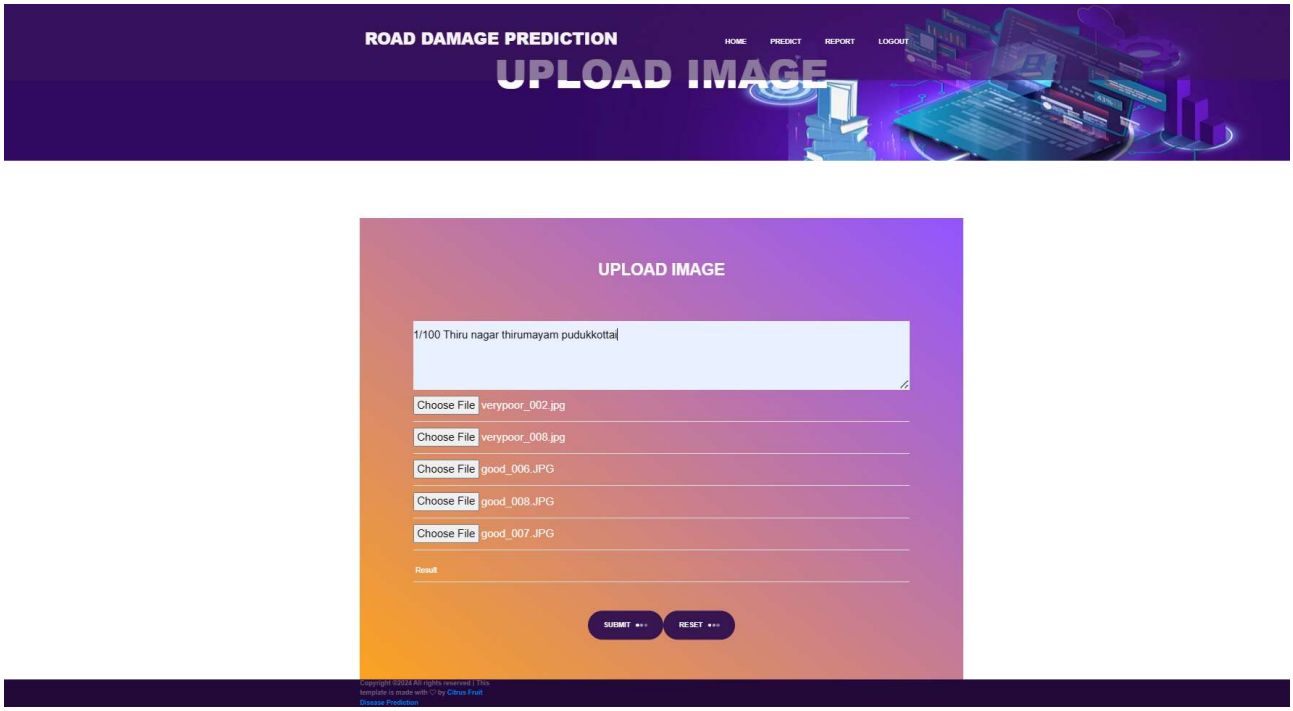

ROAD DAMAGE PREDICTION
HOME
PREDICT
REPORT
LOGOUT
UPLOAD IMAGE


UPLOAD IMAGE
Location
Choose File No file chosen
Choose File No file chosen
Choose File No file chosen
Choose File No file chosen
Choose File No file chosen
Result
SUBMIT
RESET

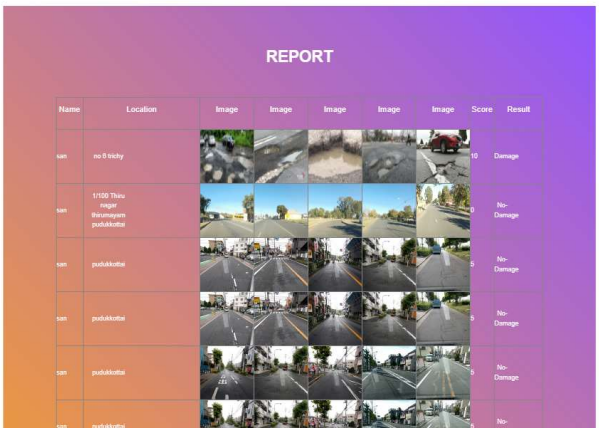

Open
Data › very_poor
Search very_poor
Organize
New folder
OneDrive - Perso
Desktop
Documents
Pictures
verypoor_001
verypoor_002
verypoor_003
verypoor_004
verypoor_005
verypoor_006
verypoor_007
verypoor_008
File name:
All files
Upload from mobile
Open
Cancel
HOME
PREDICT
REPORT
LOGOUT
Choose File No file chosen
Choose File No file chosen
Choose File No file chosen
Result
SUBMIT
RESET

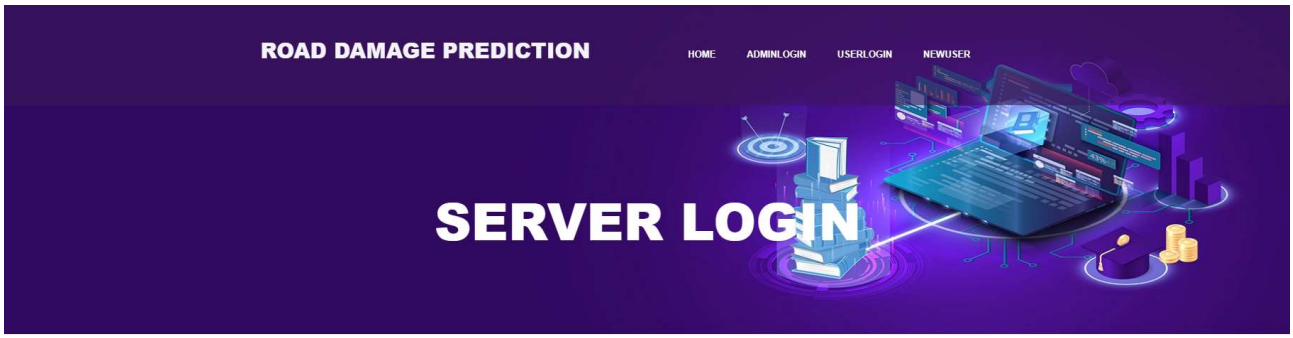

ROAD DAMAGE PREDICTION
HOME
PREDICT
REPORT
LOGOUT
UPLOAD IMAGE
UPLOAD IMAGE
Location
Choose File verypoor_002.jpg
Choose File verypoor_008.jpg
Choose File good_006.JPG
Choose File good_008.JPG
Choose File good_007.JPG
Result
SUBMIT
RESET

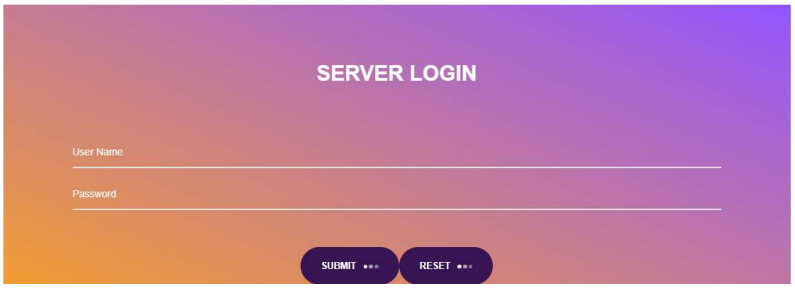

ROAD DAMAGE PREDICTION
HOME
PREDICT
REPORT
LOGOUT
UPLOAD IMAGE
UPLOAD IMAGE
Location
Choose File verypoor_002.jpg
Please fill out this field.
Choose File verypoor_008.jpg
Choose File good_006.JPG
Choose File good_008.JPG
Choose File good_007.JPG
Result
SUBMIT
RESET

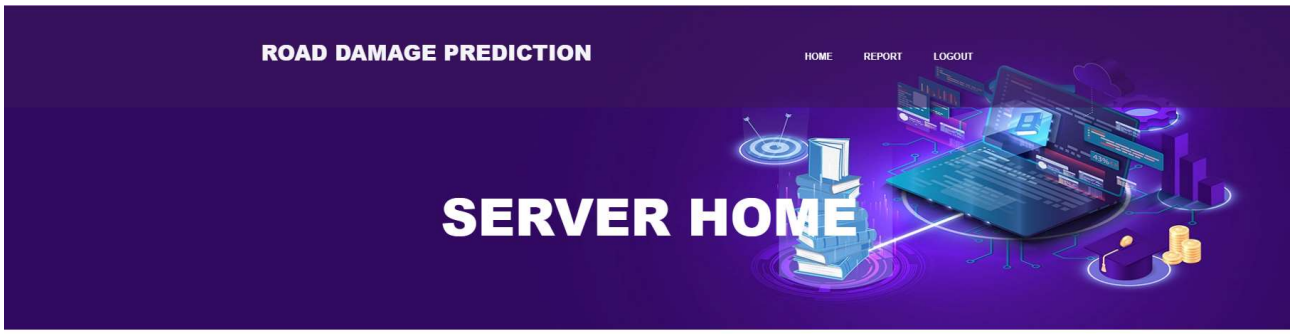

ROAD DAMAGE PREDICTION
HOME
PREDICT
REPORT
LOGOUT
UPLOAD IMAGE
UPLOAD IMAGE
1/100 Thiru nagar thirumayam pudukkottai
Choose File verypoor_002.jpg
Choose File verypoor_008.jpg
Choose File good_006.JPG
Choose File good_008.JPG
Choose File good_007.JPG
Result
SUBMIT
RESET


ROAD DAMAGE PREDICTION
HOME
PREDICT
REPORT
LOGOUT
USER HOME


REPORT
Name
Location
Image
Image
Image
Image
Image
Score
Result
no 8 trichy
Damage
1/100 Thiru nagar thirumayam pudukkottai
No-Damage
pudukkottai
No-Damage
pudukkottai
No-Damage
pudukkottai
No-Damage

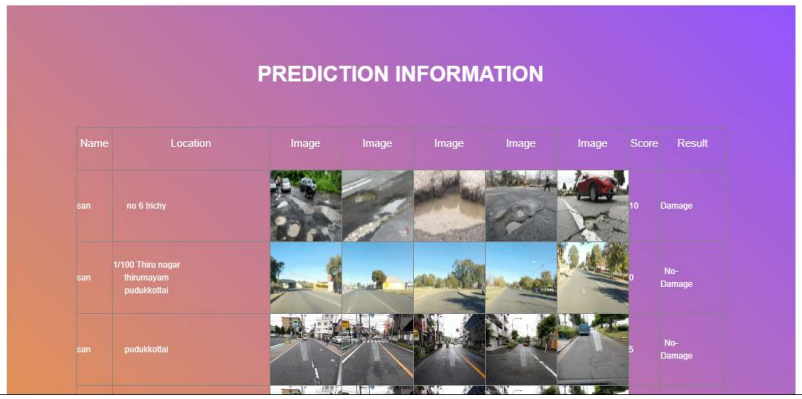

ROAD DAMAGE PREDICTION
HOME
ADMINLOGIN
USERLOGIN
NEWUSER
SERVER LOGIN


SERVER LOGIN
User Name
Password
SUBMIT
RESET


ROAD DAMAGE PREDICTION
HOME
REPORT
LOGOUT
SERVER HOME


USER INFORMATION
Name
Mobile
Email
Address
UserName
sangeeth Kumar
9486365535
sangeeth5535@gmail.com
No 16, Samnath Plaza, Madurai Main Road, Melapudhur
san
Copyright ©2024 All rights reserved | This template is made with ♡ by Road Damage


ROAD DAMAGE PREDICTION
HOME
REPORT
LOGOUT
OWNER HOME


PREDICTION INFORMATION
Name
Location
Image
Image
Image
Image
Image
Score
Result
san
no 6 trichy
10
Damage
san
1/100 Thiru nagar
thirumayam
pudukkottai
No-
Damage
san
pudukkottai
No-
Damage

REFERENCES

[1] Shu, Jiangpeng, et al. "An active learning method with difficulty learning mechanism for crack detection." Smart Struct Syst 29.1 (2022): 195-206.

[2] Yang, Yalong, et al. "Research on Pavement Crack Detection Algorithm based on Deep Residual Unet Neural Network." Journal of Physics: Conference Series. Vol. 2278. No. 1. IOP Publishing, 2022.

[3] Vrochidou, Eleni, et al. "Towards Robotic Marble Resin Application: Crack Detection on Marble Using Deep Learning." Electronics 11.20 (2022): 3289.

[4] Golding, Vaughn Peter, et al. "Crack Detection in Concrete Structures Using Deep Learning." Sustainability 14.13 (2022): 8117.

[5] Ren, Junhua, et al. "Automatic Pavement Crack Detection Fusing Attention Mechanism." Electronics 11.21 (2022): 3622.

[6] Ramesh, Akshatha, et al. "Cloud-based collaborative road-damage monitoring with deep learning and smartphones." Sustainability 14.14 (2022): 8682.

[7] Ranyal, Eshta, Ayan Sadhu, and Kamal Jain. "Road condition monitoring using smart sensing and artificial intelligence: A review." Sensors 22.8 (2022): 3044.

[8] Katsaliros, Aggelos, et al. "Road Crack Detection Using Quaternion Neural Networks." 2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP). IEEE, 2022.

[9] Branikas, Efstathios, Paul Murray, and Graeme West. "A novel data augmentation method for improved visual crack detection using generative adversarial networks." IEEE Access 11 (2023): 22051-22059.

[10] Pantoja-Rosero, Bryan G., et al. "TOPO-Loss for continuity-preserving crack detection using deep learning." Construction and Building Materials 344 (2022): 128264.