GAN AUGMENTED VISION TRANSFORMER APPROACH FOR DETECTING CRACKS IN CONCRETE

A PROJECT REPORT

Submitted by

MURUGESHWARI A	**211420243035**
PRIYADHARSHINI S	**211420243040**
SIVASANKARI K	**211420243052**

in partial fulfilment for the award of degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **“GAN AUGMENTED VISION TRANSFORMER APPROACH FOR DETECTING CRACKS IN CONCRETE”** is the bonafide work of **MURUGESHWARI A [REGISTER NO: 211420243035], PRIYADHARSHINI S [REGISTER NO: 211420243040], SIVASANKARI K [REGISTER NO: 211420243052]** who carried out this project work under **Mrs. R. PRIYA** supervision.

SIGNATURE OF THE HOD	**SIGNATURE OF THE SUPERVISOR**
Dr. S. MALATHI M.E., Ph.D.,	**Mrs. R. PRIYA**
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR**
DEPARTMENT OF AI&DS,	DEPARTMENT OF AI&DS,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR ENGINEERING COLLEGE,
CHENNAI-600123.	CHENNAI-600123.

Certified that the above mentioned students were examined in End Semester project (AD8811) viva-voice held on ……………………

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **Murugeshwari A (211420243035), Priyadharshini S (211420243040), Sivasankari K (211420243052)** hereby declare that this project report titled "**GAN Augmented Vision Transformer approach for detecting crack in concrete**" under the guidance of **Mrs. R. Priya** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. S. MALATHI, M.E., Ph.D.,** Head of the Department of Artificial Intelligence and Data Science, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Project Coordinator Dr. K. JAYASHREE** and **Project Guide Mrs. R. PRIYA** and all the faculty members of the Department of AI&DS for their unwavering support for the successful completion of the project.

MURUGESHWARI A

PRIYADHARSHINI S

SIVASANKARI K

ABSTRACT

This research introduces a novel approach for detecting cracks in concrete using a Generative Adversarial Network (GAN)-augmented Vision Transformer (ViT) model. The proposed methodology leverages the power of GANs to enhance the dataset, addressing the limited availability of diverse crack patterns. By generating synthetic crack images, the model is trained on a more comprehensive and representative dataset. The Vision Transformer architecture is employed to capture long-range dependencies in the concrete images, enabling the detection of subtle crack formations. Experimental results demonstrate the effectiveness of the GAN-augmented ViT approach, showcasing improved crack detection accuracy compared to traditional methods. This innovative fusion of GANs and Vision Transformers contributes to advancing the reliability and robustness of crack detection systems in concrete structures.

KEY WORDS: Generative Adversarial Network (GAN), Vision Transformer (ViT), crack detection, synthetic data augmentation, long-range dependencies, image classification, dataset diversity.

LIST OF FIGURES

LIST OF ABBREVATIONS

S.NO	ABBREVATION	EXPANSION
1	GAN	Generative Adversarial Network
2	VIT	Vision Transformer
3	CNN	Convolutional Neural Network
4	RGB	Red Green Blue
5	ReLU	Rectified Linear Unit
6	ML	Machine Learning
7	DL	Deep Learning
8	AI	Artificial Intelligence
9	VS Code	Visual Studio Code
10	DFD	Data Flow Diagram
11	UML	Unified Modeling Language
12	ERD	Entity Relationship Diagram
13	METU	Middle East Technical University
14	HTML	Hyper Text Markup Language
15	CV	Computer Vision
16	MLP	Multilayer Perceptron
17	MSE	Mean Squared Error
18	MAE	Mean Absolute Error

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Concrete is a fundamental material in construction, relied upon for its durability and strength. However, over time, concrete structures are subject to various forms of degradation, with cracks being a common issue. Detecting and monitoring cracks in concrete is crucial for ensuring structural integrity and safety. Traditional methods for crack detection often rely on manual inspection, which can be time-consuming, labor-intensive, and prone to human error. Moreover, with the increasing complexity and scale of infrastructure projects, there is a growing need for automated and efficient crack detection systems.Recent advancements in artificial intelligence, particularly in the field of computer vision, have shown promise in automating crack detection processes. Generative Adversarial Networks (GANs) have emerged as powerful tools for generating realistic images, which can be utilized to augment datasets and enhance the performance of computer vision models. Our proposed approach integrates the strengths of GANs for data augmentation and ViTs for feature extraction and classification. By synthesizing additional realistic crack images using GANs, we aim to mitigate the challenge of limited annotated data, which is often a bottleneck in training robust deep learning models. Furthermore, the Vision Transformer architecture enables efficient processing of high-resolution images, allowing us to capture intricate details and patterns associated with cracks in concrete. The self-attention mechanism of ViTs facilitates learning global and local features, enabling the model to discern subtle cracks amidst complex backgrounds and lighting conditions. The research proposes a novel methodology that harnesses the capabilities of both GANs and Vision Transformers to generalize to unseen data.Concrete structures are susceptible to cracks over time due to various factors such as environmental conditions, material properties, and structural loads.

1.1 OVERVIEW

The proposed research focuses on addressing the challenge of automated crack detection in concrete structures using advanced machine learning techniques. Specifically, the approach combines Generative Adversarial Networks (GANs) and Vision Transformers (ViTs) to enhance the accuracy, efficiency, and robustness of crack detection systems. Concrete structures are susceptible to cracks over time due to various factors such as environmental conditions, material properties, and structural loads. Manual inspection for crack detection is labor-intensive and error-prone, necessitating the development of automated solutions.The integration of artificial intelligence, particularly computer vision, offers promising avenues for automating crack detection processes. However, existing methods often face challenges such as limited annotated data and the complexity of crack patterns in real-world scenarios. The research proposes a novel methodology that harnesses the capabilities of both GANs and Vision Transformers to generalize to unseen data.

1.2 PROBLEM DEFINITION

The problem addressed in this research is the automated detection of cracks in concrete structures using a GAN-augmented Vision Transformer approach. The key challenges associated with this problem includes Annotated datasets for crack detection in concrete structures are often limited in size and diversity, hindering the training of robust machine learning models.Cracks in concrete can exhibit diverse shapes, sizes, orientations, and textures, making them challenging to detect accurately using conventional computer vision techniques. The developed model must generalize well to diverse environmental conditions, lighting variations, surface textures, and crack severities encountered in real-world scenarios. Cracks in solar modules can exhibit complex shapes, orientations, and textures, making accurate segmentation and feature extraction challenging for automated algorithms.

Transformers for feature extraction and classification requires careful design and optimization to leverage the complementary strengths of both techniques effectively.The transition of the developed model from research to practical deployment involves considerations such as hardware requirements, deployment costs, and integration with existing infrastructure monitoring systems. The automated crack detection system must be reliable and accurate to ensure the safety and structural integrity of concrete infrastructure. False positives or false negatives could lead to erroneous maintenance decisions or, worse, compromise structural safety. Addressing these challenges effectively is essential for developing a robust, efficient, and deployable solution for automated crack detection in concrete structures using the proposed GAN-augmented Vision Transformer approach.

1.3 OBJECTIVE

Generative Adversarial Networks (GANs) and Vision Transformers (ViTs) for crack detection in concrete structures. This involves optimizing the network architecture, training procedures, and hyperparameters to effectively leverage the strengths of both GANs and ViTs.

1.4 SCOPE

Further refinement of the GAN architecture for generating synthetic crack images, including exploration of different GAN variants and training strategies to improve realism and diversity in generated images. Optimization of Vision Transformer architecture for crack detection, considering variations in model depth, attention mechanisms, and input image resolution to achieve optimal performance. Investigation of additional data augmentation techniques beyond GANs, such as geometric transformations, color jittering, and style transfer, to further diversify the training dataset and improve model generalization

CHAPTER 2

LITERATURE SURVEY

2.1 Automatic Crack Segmentation and Feature Extraction in Electroluminescence Images of Solar Modules

AUTHORS - Xin Chen , Todd Karin , Cara Libby

YEAR – 2023

This paper proposes a CNN-based approach for crack detection in solar panels using visible light images. It discusses the use of transfer learning and data augmentation techniques to improve model performance. A CNN-based method for automatic defect recognition in solar panels, including crack detection. The study focuses on optimizing CNN architectures and evaluating the model's performance on real-world datasets. explore the application of deep learning techniques, specifically CNNs, for crack detection in concrete structures using infrared thermography images. The study discusses the challenges and opportunities in utilizing thermal imaging for crack detection.

MERITS - Automation of crack segmentation and feature extraction reduces the need for manual inspection, saving time and labor costs in solar module quality assessment.

DEMERITS - Cracks in solar modules can exhibit complex shapes, orientations, and textures, making accurate segmentation and feature extraction challenging for automated algorithms.

2.2 Analysis of PV module power loss and cell crack effects due to accelerated aging

AUTHORS- C. Libby et al

YEAR-2023

The impact of accelerated aging tests and field exposure on the power loss and crack formation in photovoltaic (PV) modules. The study aims to provide insights into the performance degradation mechanisms of PV modules under different environmental conditions and aging scenarios. Accelerated aging tests serve as a controlled environment for simulating the effects of prolonged exposure to environmental stressors such as temperature, humidity, and ultraviolet (UV) radiation. Conversely, field exposure studies provide insights into the real-world performance of PV modules under actual operating conditions, including varying weather patterns, solar irradiance levels, and geographical locations. By quantitatively assessing power loss and cell crack effects, the study seeks to identify key factors contributing to PV module degradation and provide practical insights for improving module design, reliability testing protocols, and maintenance practices.

MERITS - Analysis of photovoltaic (PV) module power loss and the effects of cell cracks resulting from both accelerated aging tests and field exposure. This comprehensive approach provides valuable insights into the performance degradation of PV modules over time.

DEMERITS - The reliability of accelerated aging tests may be sensitive to the specific test conditions employed, including temperature, humidity, and irradiance levels. Variations in test parameters could impact the accuracy of predictions regarding real-world performance degradation.

2.3 Impact of cracks in multi crystalline silicon solar cells on PV module power a simulation study based on field data

AUTHORS- A.Morlier , F.Haase and M.Köntges

YEAR-2019

The paper under survey explores the impact of cracks in multi-crystalline silicon solar cells on the overall power output of photovoltaic (PV) modules. As cracks are a common occurrence in solar cells, understanding their effects on module performance is crucial for optimizing solar energy systems' reliability and efficiency.

Cracks in solar cells can arise from various sources, including manufacturing defects, installation stress, thermal cycling, and mechanical damage. These cracks can propagate over time, leading to reduced electrical conductivity, increased recombination losses, and localized shading effects, ultimately impacting the module's power output. The study employs simulation-based analysis, leveraging field data to model the effects of cracks on PV module performance under real-world operating conditions.

MERITS: By elucidating the impact of cracks on PV module power output, the paper offers practical insights for system designers, installers, and operators. This information can inform decisions related to module selection, installation practices, and maintenance strategies aimed at minimizing power loss due to cracks.

DEMERITS: The accuracy of the simulation results hinges on the accurate detection and characterization of cracks in the field data. Challenges in crack detection methods or limitations in the resolution of imaging techniques could introduce uncertainties into the analysis.

2.4 Deep learning-based automatic detection of multitype defects in photovoltaic modules and application in real production line

AUTHORS- Y. Zhao, K. Zhan, Z. Wang, and W. Shen
YEAR-2021

The application of deep learning techniques for the automatic detection of multiple types of defects in photovoltaic (PV) modules, with a focus on real-world production line scenarios. As the demand for solar energy continues to rise, ensuring the quality and reliability of PV modules is essential for maximizing energy production and prolonging system lifespan. The detection of defects in PV modules traditionally relies on manual inspection, which can be labor-intensive, subjective, and prone to errors. By leveraging deep learning algorithms, the study aims to automate the defect detection process, enabling faster, more accurate, and more consistent identification of various types of defects. They examine the methodology, findings, strengths, and limitations of the paper, assessing its contributions to the field of solar energy research and its practical implications for enhancing quality control in PV module manufacturing.

MERITS: Automated defect detection in PV module production lines, leveraging deep learning algorithms to streamline the inspection process. Automation improves efficiency and reduces labor costs while enhancing the consistency and reliability of defect identification.

DEMERITS: Annotating large datasets of PV module images with diverse defect types for training deep learning models can be time-consuming and labor-intensive. Ensuring the quality and consistency of annotations is crucial for the effectiveness of the defect detection system.

2.5 Segmentation of photovoltaic module cells in uncalibrated electroluminescence images

AUTHORS- S. Deitsch

YEAR-2021

The segmentation of photovoltaic (PV) module cells in uncalibrated electroluminescence (EL) images, aiming to automate and enhance the efficiency of solar cell analysis. Electroluminescence imaging is a non-destructive technique used for assessing the quality and integrity of solar cells within PV modules, offering insights into defects, cracks, and performance variations. Segmentation of individual solar cells in EL images is a critical step in the analysis process, enabling quantitative measurements of cell characteristics such as size, shape, and spatial distribution. However, segmentation in EL images presents unique challenges due to variations in lighting, noise levels, and irregularities in cell boundaries. A segmentation algorithm tailored to uncalibrated EL images, leveraging techniques from image processing and computer vision.

MERITS: The paper employs a deep learning-based approach for cell segmentation, leveraging convolutional neural networks (CNNs) to automatically learn relevant features from EL images. Deep learning methods have demonstrated superior performance in image segmentation tasks, offering robustness to variations in image characteristics.

DEMERITS: Training deep learning models for cell segmentation entails significant computational resources, including high-performance GPUs and large-scale parallel processing. The computational complexity may limit the feasibility of deploying the segmentation method in resource-constrained environments or real-time applications.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The CNN starts with an input layer that takes RGB images of concrete surfaces as input. These images are then passed through several convolutional layers, each followed by a rectified linear unit (ReLU) activation function. These layers extract features from the images at different levels of abstraction, allowing the network to learn hierarchical representations of cracks in the concrete. Pooling layers, such as max pooling or average pooling, are used to downsample the feature maps, reducing the spatial dimensions while retaining important features. This helps in reducing the computational complexity of the network and improving its ability to generalize to new, unseen images. The output of the convolutional layers is flattened into a one-dimensional vector, which is then passed through one or more fully connected layers. These layers further process the features and map them to the output classes, which in this case are "crack" and "no crack". The fully connected layers typically use the ReLU activation function to introduce non-linearity into the model. The final layer of the CNN is the output layer, which uses a sigmoid activation function to produce a probability score for each class. During training, the network is optimized using a labeled dataset of images with cracks and without cracks. Once trained, the CNN can be evaluated on a separate test set to assess its performance using metrics such as accuracy, precision, recall, and F1 score.

DEMERITS

- Requires large amounts of labeled data for training.
- Prone to overfitting, especially with complex architectures.
- Computationally intensive, especially with large image datasets.
- May struggle with detecting cracks in varying lighting conditions.

3.2 PROPOSED SYSTEM

In the proposed GAN-augmented Vision Transformer (ViT) approach for crack detection in concrete, the model consists of three main components: the Generator (G), the Discriminator (D), and the Vision Transformer (ViT). The Generator takes a latent representation or random noise vector as input and generates synthetic crack images. These synthetic images are then fed into the Discriminator along with real crack images. This feature representation is crucial for the overall model's performance in distinguishing between cracked and uncracked areas in concrete. The ViT can be pretrained on a large dataset to learn general features and then fine-tuned on the specific task of crack detection using both real and synthetic images. During training, the model undergoes alternating optimization. This adversarial training process helps the Generator improve its ability to create realistic synthetic images, which in turn enhances the ViT's performance in crack detection. The loss functions used in training include binary cross-entropy for both the Discriminator and the Generator. Once trained, the model can be evaluated on a separate validation set using standard metrics such as accuracy, precision, recall, and F1 score. This evaluation helps assess the model's performance in detecting cracks in concrete surfaces and its generalization ability to unseen data. Overall, the GAN-augmented ViT approach offers a promising method for improving crack detection in concrete by leveraging the power of both GANs and transformers in image processing tasks.

MERITS

- Enhanced performance through synthetic image generation.
- Improved model robustness with augmented dataset.
- Ability to learn intricate crack patterns from synthetic data.
- Potential for reducing reliance on large labeled datasets.
- Adaptability to different lighting and environmental conditions.

3.3 FESIBILITY STUDY

3.3.1 TECHNICAL FEASIBILITY

This The proposed GAN-augmented Vision Transformer (ViT) approach for crack detection in concrete appears technically feasible. GANs have been successfully applied in image generation tasks, and ViTs have shown promise in various computer vision tasks. Integrating these techniques for crack detection can potentially enhance the model's performance. ViTs, with their self-attention mechanism, can capture long-range dependencies in images, which is beneficial for detecting cracks that may span large areas. GANs can generate synthetic images to augment the dataset, addressing the challenge of limited labeled data. This approach also aligns with the trend of using transformer-based models for image processing tasks, indicating its technical viability. However, challenges such as training complexity and hyperparameter tuning need to be addressed to ensure the model's effectiveness. Overall, with proper implementation and optimization, this approach holds technical promise for crack detection in concrete.

Why PYTHON is used?

Python is an interpreted high-level general purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms. It is often described as a "batteries included" language due to its comprehensive standard library. Python's simplicity and consistency, as well as access to excellent libraries and frameworks for Al and machine learning (ML), flexibility, platform freedom, and a large community, make it the best choice for machine learning and Al applications.

3.3.1.1 REQUIREMENTS

VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a popular, free, and open-source code editor developed by Microsoft. It is widely used by developers for various programming languages and platforms due to its versatility, ease of use, and extensive set of features. One of its key strengths is its lightweight nature, making it fast and responsive even when working with large codebases. VS Code offers a wide range of extensions that enhance its functionality, allowing users to customize their coding experience to suit their preferences and workflow. One of the standout features of VS Code is its built-in Git integration, which provides seamless version control directly within the editor. This makes it easy for developers to manage their code changes and collaborate with others. Additionally, VS Code offers a powerful debugging experience with support for breakpoints, watch variables, and interactive debugging sessions. Another notable feature is the IntelliSense autocomplete functionality, which provides intelligent code suggestions as you type, helping to speed up coding and reduce errors. VS Code also includes a built-in terminal, so developers can run commands and scripts without leaving the editor, streamlining their workflow.

3.3.2 ECONOMICAL FEASIBILITY

The economic feasibility of implementing a GAN-augmented Vision Transformer (ViT) approach for crack detection in concrete depends on several factors. Initially, there may be higher upfront costs associated with developing and fine-tuning the model, as well as acquiring or generating a dataset of concrete images. However, once the model is trained and deployed, ongoing costs are minimal. The use of synthetic data generated by the GAN can reduce the reliance on expensive, labor-intensive data collection and labeling processes, potentially lowering overall costs.

3.3.3 SOCIAL FEASIBILITY

The social feasibility of implementing a GAN-augmented Vision Transformer (ViT) approach for crack detection in concrete involves several considerations. Firstly, there may be concerns regarding the accuracy and reliability of the model, especially in critical infrastructure such as buildings and bridges where safety is paramount. Ensuring that the model performs reliably and consistently in real-world is crucial to gaining acceptance from stakeholders and the public. Furthermore, there may be resistance from traditionalists within the industry who are skeptical of adopting new technologies, especially in fields such as civil engineering where established practices and standards are highly valued. Educating and engaging with stakeholders to demonstrate the benefits and reliability of the GAN-augmented ViT approach would be essential to gaining acceptance and support. the social feasibility of this approach would depend on factors such as the level of trust in the technology, the willingness of stakeholders to adopt new methods, and the ability to address ethical and social concerns effectively.

3.4 HARDWARE REQUIREMENTS

• Processor	- Intel i3 or later
• Hard disk	- minimum 10 GB
• RAM	- minimum 4 GB

3.5 SOFTWARE REQUIREMENTS

• Operating System	- Windows 10 or later
• Tool	- Visual Studio Code
• Frontend and Backend	- HTML, CSS
• Language	- Python

CHAPTER 4

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

4.1.1 LEVEL 0

Level 0 describe the whole process of the project. We can use data set as an input. The system will use GAN Augmentation and VIT Module to predict crack.

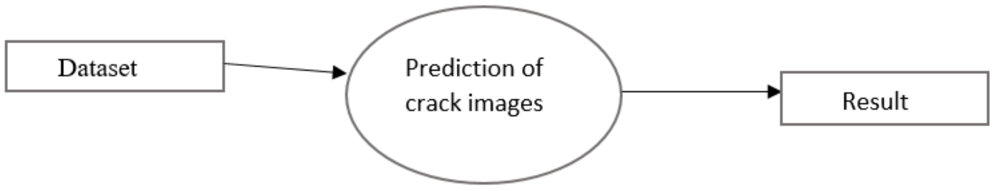


Fig No 4.1 Level 0 data flow diagram

4.1.2 LEVEL 1

Level 1 describes the preliminary consideration and process of extracting a project features. Weather data is given as input and the data is processed in advance. The preprocessing phase involves extracting records with empty values.

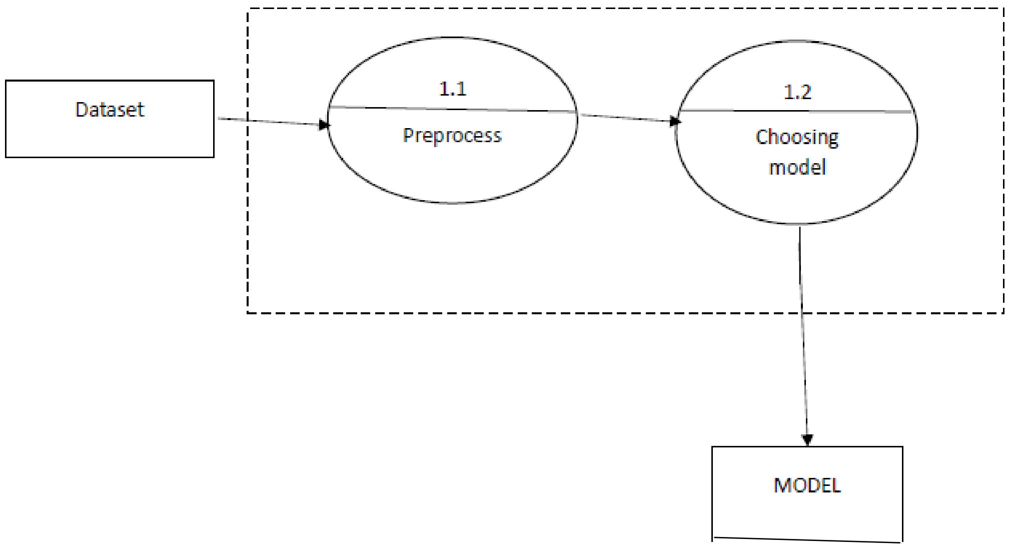


Fig No. 4.2 Level 1 data flow diagram

4.1.3 LEVEL 2

Level 2 describes a more detailed process which is the flow of project work. Once the database is trained it will read historical data and user data and predict power and radiation. By using appropriate model, you will get the prediction.

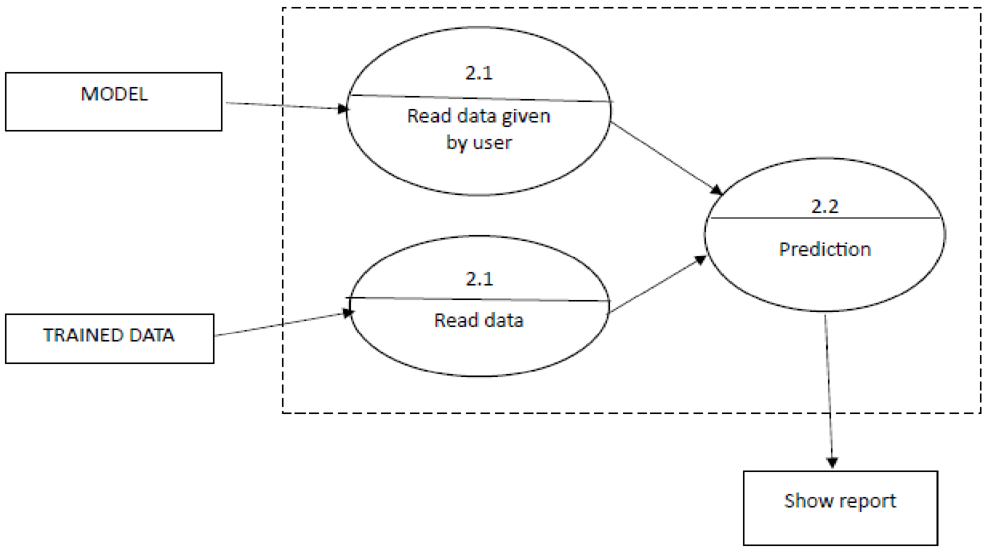


Fig No. 4.3 Level 2 data flow diagram

4.2 UML DIAGRAM

4.2.1 USE CASE DIAGRAM

Use case diagrams are considered for high level requirement analysis of a system. A simple use case diagram represents the user interaction with a system that shows the relationship between the user and different operating conditions in which the user is involved. When the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

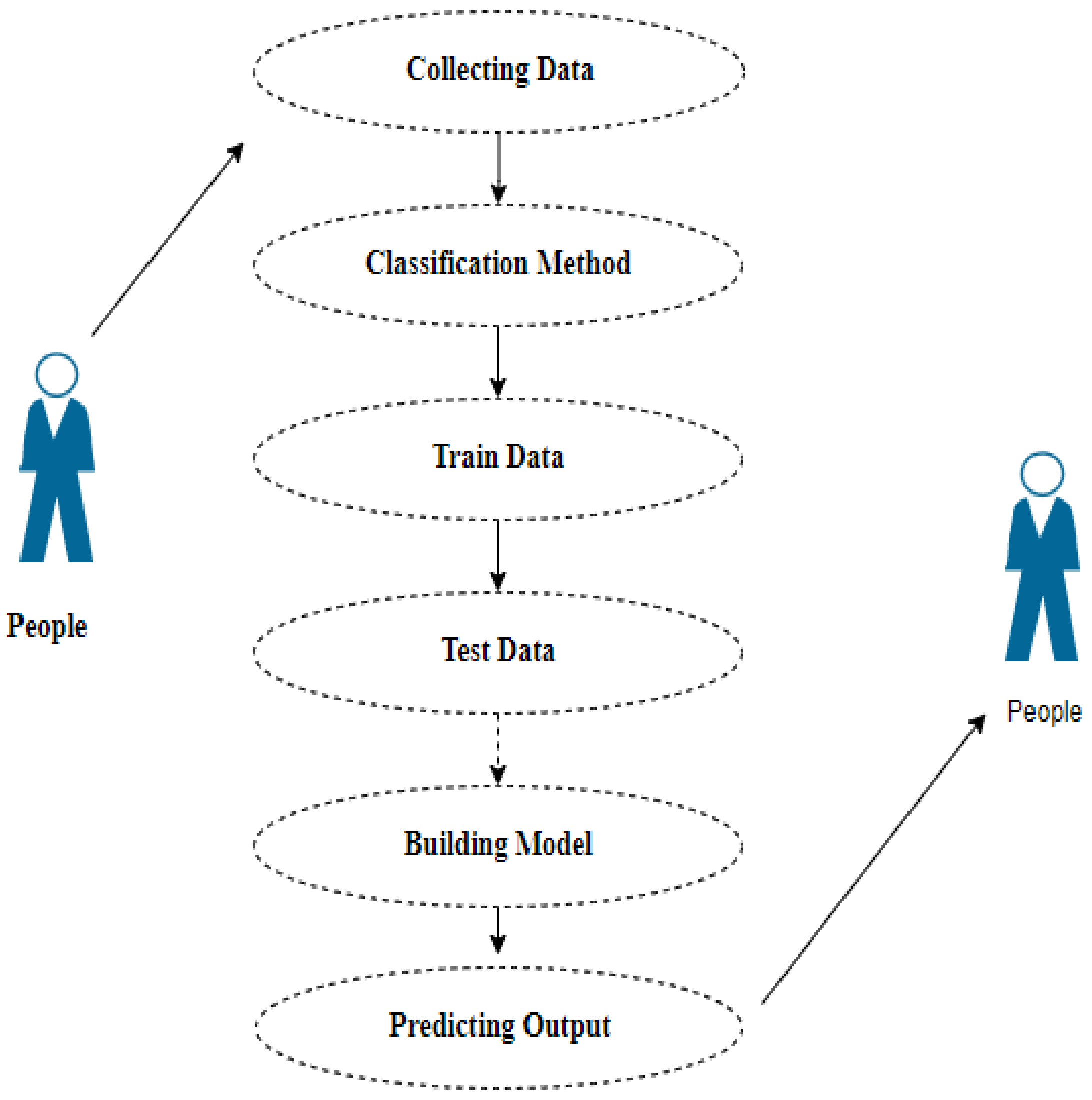


Fig No. 4.4 Use case Diagram

4.2.2 CLASS DIAGRAM

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated.

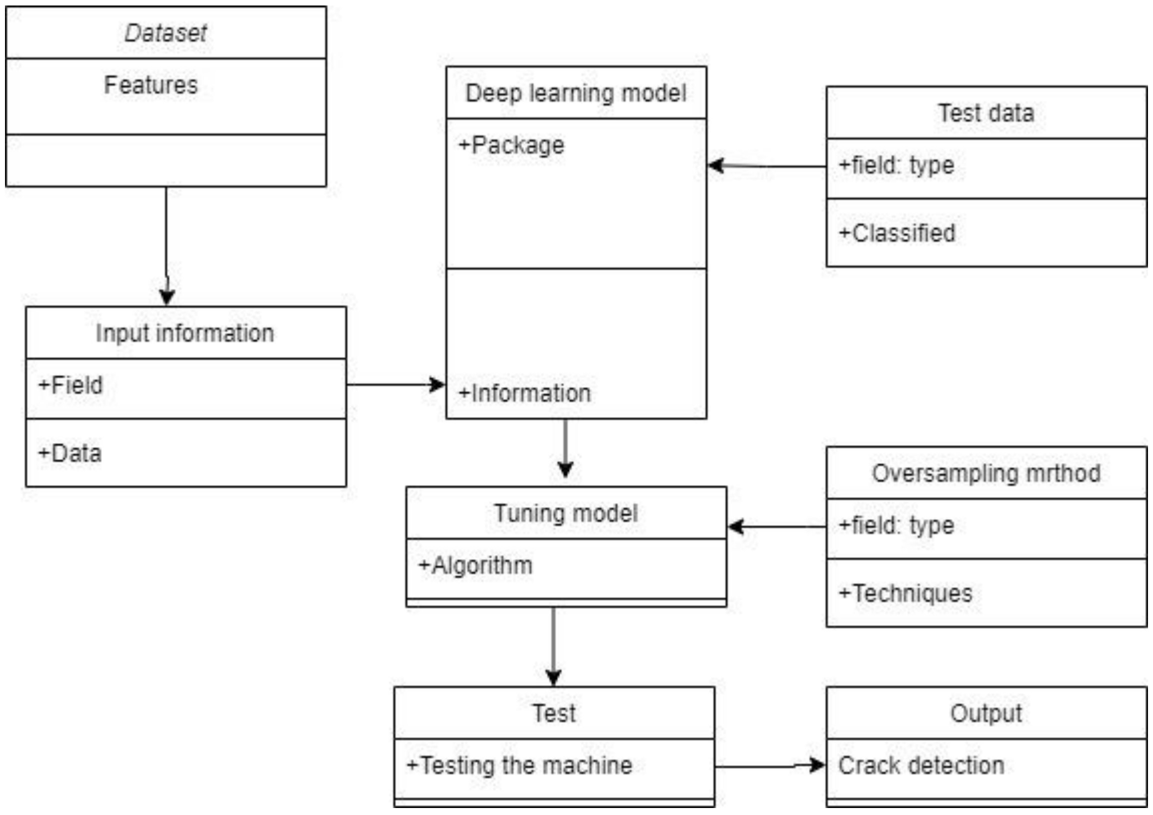


Fig No. 4.5 Class Diagram

4.2.3 SEQUENCE DIAGRAM

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Sequence diagrams, along with class diagrams and physical data models are the most important design-level models for modern business application development.

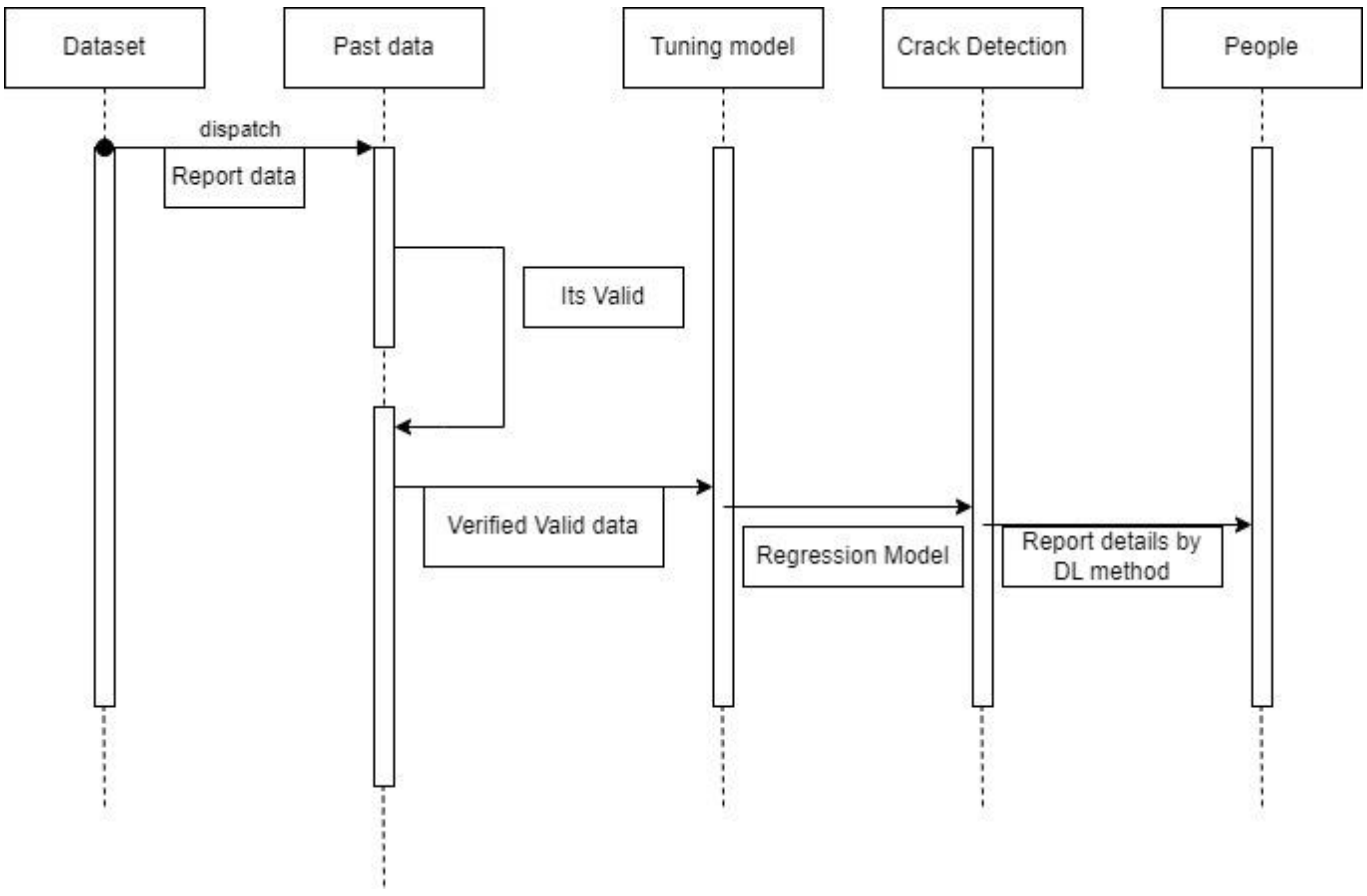


Fig No. 4.6 Sequence Diagram

4.2.4 ACTIVITY DIAGRAM

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams look like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

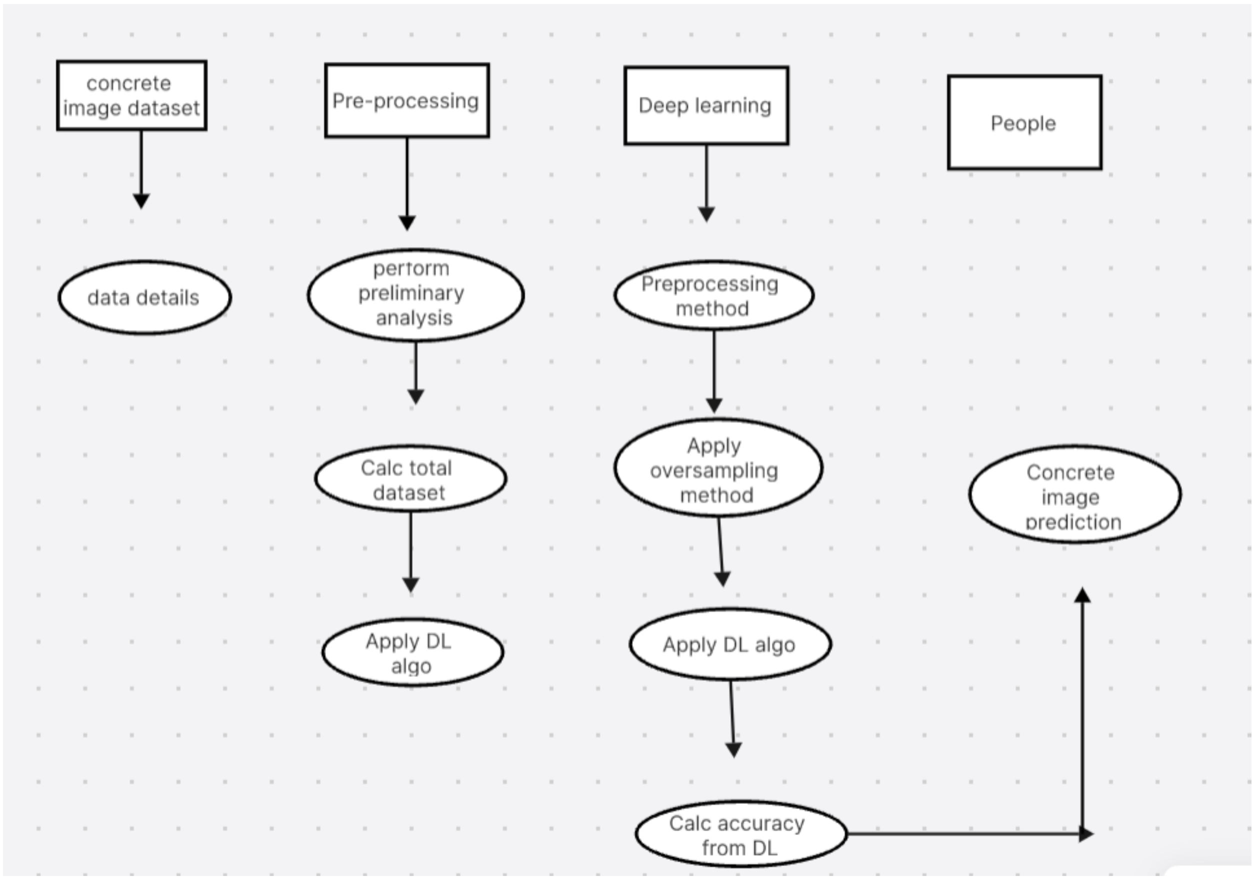


Fig No. 4.7 Activity Diagram

4.3 ENTITY RELATIONSHIP DIAGRAM(ERD)

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

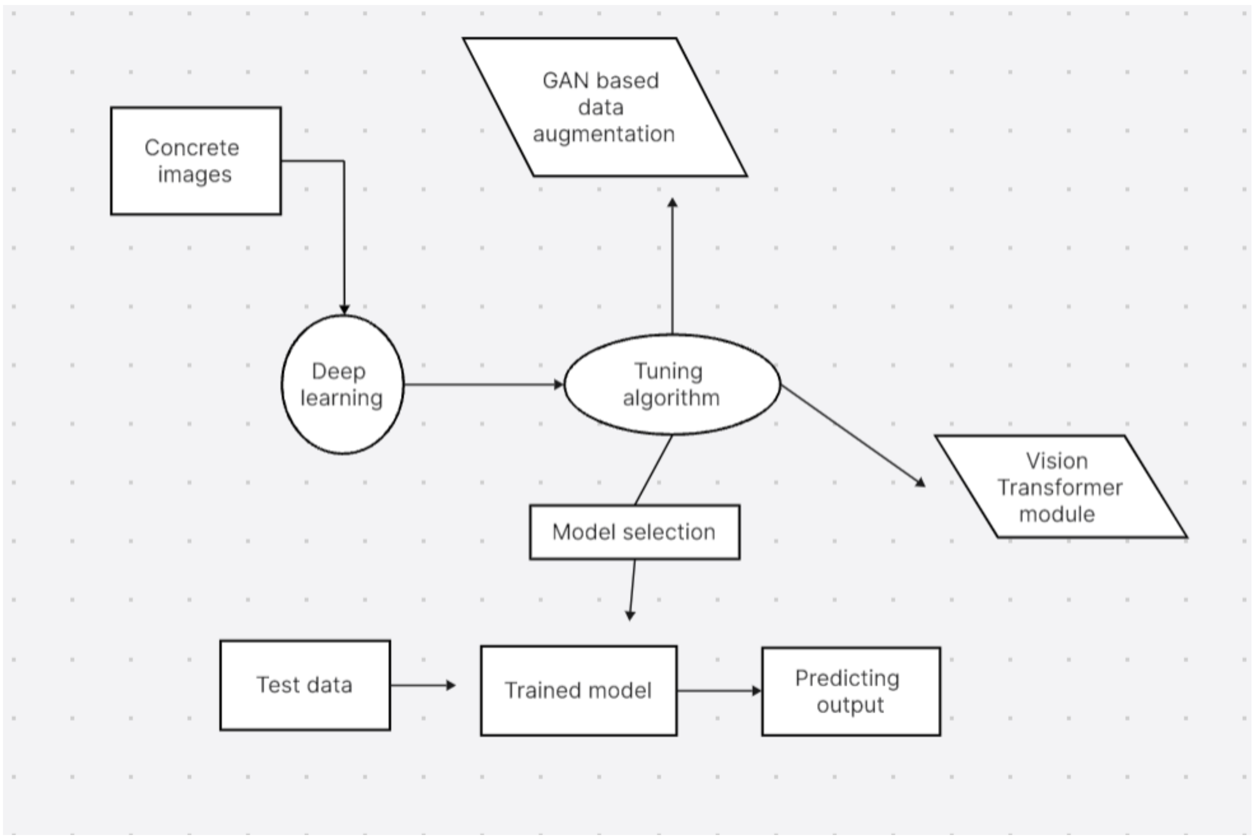


Fig No. 4.8 Entity Relationship Diagram

CHAPTER 5
SYSTEM ARCHITECTURE

The proposed framework offers an innovative and integrated approach to crack detection, harnessing the synergies between Generative Adversarial Network (GAN) augmentation and Vision Transformer (ViT) technology. By combining these advanced techniques, the framework aims to enhance the precision, reliability, and robustness of crack detection systems, thereby addressing the challenges associated with traditional methods and paving the way for more effective infrastructure maintenance and safety protocols. Acquiring a high-quality dataset that accurately represents the complexities of real-world scenarios is crucial for training a robust crack detection model.

In this project, the dataset consists of concrete images with cracks sourced from various METU Campus Buildings. The dataset is curated to ensure diversity and variability, capturing a wide range of environmental conditions, lighting scenarios, and types of cracks commonly encountered in concrete structures. The dataset is meticulously divided into two categories: negative and positive crack images, facilitating image classification tasks. Each class comprises 20,000 images, resulting in a total of 40,000 images, all standardized to dimensions of 227 x 227 pixels with RGB channels. These images are extracted from 458 high-resolution images, each measuring 4032x3024 pixels, employing the methodology proposed by Zhang et al. (2016).Additionally, synthetic data is also generated using Generative Adversarial Networks (GANs) to further enhance the diversity and robustness of the dataset. This synthetic data creation process is integral to addressing the challenges associated with limited and homogeneous crack datasets, ensuring that the crack detection model is trained on a more comprehensive and representative dataset.

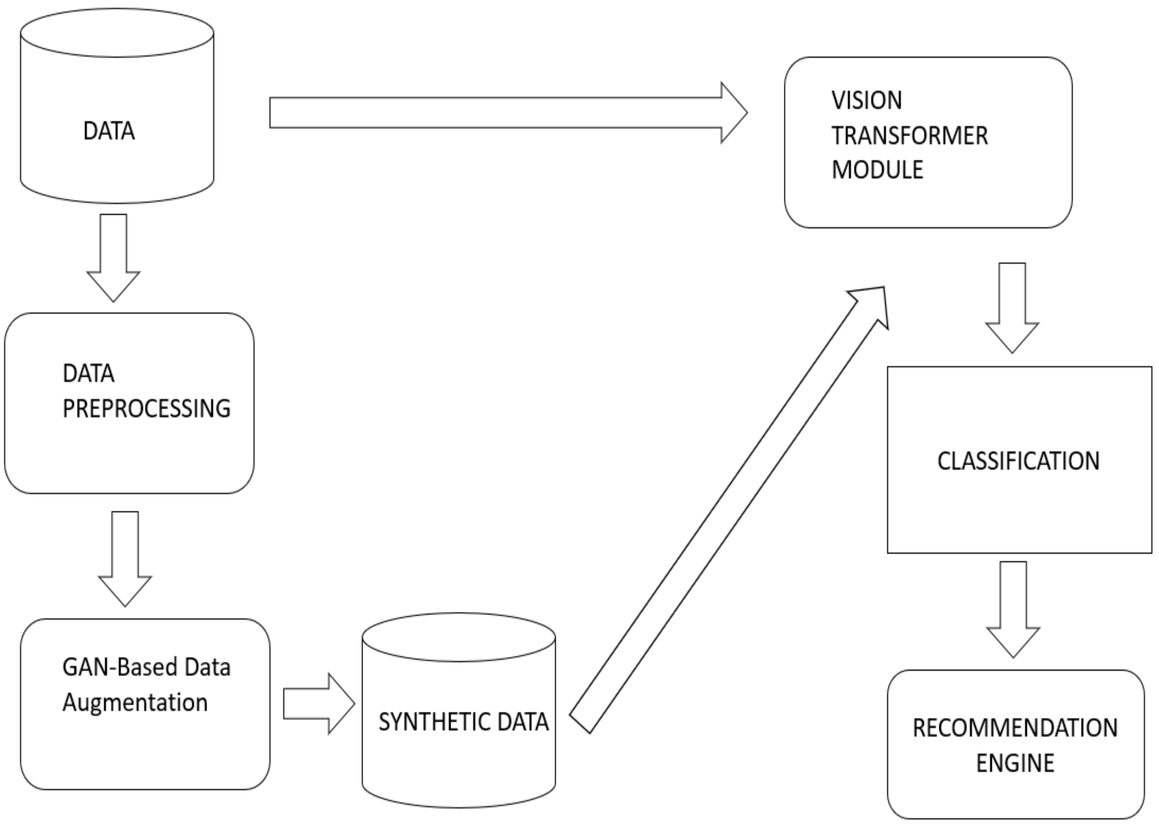


Fig No. 5.1 System Architecture

5.1MODULE DESIGN SPECIFICATION

5.1.1 LIST OF MODULES

- Data Pre-processing
- GAN based Data Augmentation
- Vision Transformer Module
- Classification
- Recommendation Engine

5.1.1.1 DATA PRE-PROCESSING

Data preprocessing is a crucial step in preparing the dataset for training a GAN-augmented Vision Transformer (ViT) model for crack detection in concrete. Several techniques can be applied to enhance the quality and usability of the data. Firstly, the images of concrete surfaces need to be properly formatted and resized to ensure consistency in input dimensions for the ViT. This involves resizing images to a uniform resolution and converting them to the appropriate color space if needed. Next, data augmentation techniques can be applied to increase the diversity of the dataset and improve the model's generalization ability. This can include random rotations, flips, and translations of the images to simulate variations in lighting and perspective. Additionally, it may be beneficial to enhance the contrast and sharpness of the images to improve the visibility of cracks. This can be achieved through techniques such as histogram equalization or adaptive contrast enhancement.To further improve the quality of the dataset, noise reduction techniques can be applied to remove unwanted artifacts or disturbances from the images. This can help the model focus on detecting actual cracks rather than being distracted by noise.Finally, it is important to carefully label the images to indicate the presence or absence of cracks. This labeling process should be accurate and consistent to ensure the model is trained on reliable ground truth data.Overall, by applying these data preprocessing techniques, the dataset can be optimized to improve the performance and robustness of the GAN-augmented ViT model for crack detection. Furthermore, it is essential to consider data imbalance issues that may arise in the dataset, as concrete surfaces with cracks may be significantly less common than those without cracks. To address this, techniques such as oversampling of minority classes or using class weights during training can be employed to ensure that the model is not biased towards the majority class. Normalization of the data is another important preprocessing step, which involves scaling the pixel values of the images to a range that is suitable for the model.

5.1.1.2 GAN BASED DATA AUGMENTATION

GAN-based data augmentation can be a powerful technique for enhancing the dataset used to train a GAN-augmented Vision Transformer (ViT) model for crack detection in concrete. The basic idea behind GAN-based data augmentation is to generate synthetic crack images that are realistic and indistinguishable from real images, thus expanding the dataset and improving the model's ability to generalize to unseen data.

To implement GAN-based data augmentation, a GAN consisting of a Generator (G) and a Discriminator (D) is trained on the existing dataset of concrete images. The Generator learns to generate synthetic crack images by mapping random noise vectors to realistic images, while the Discriminator learns to distinguish between real and synthetic images.During training, the Generator and Discriminator are trained adversarially, with the Generator trying to fool the Discriminator into classifying its synthetic images as real. This adversarial training process helps the Generator learn to generate images that closely resemble real cracks in concrete surfaces.Once the GAN is trained, the Generator can be used to generate additional synthetic crack images. These synthetic images can be added to the original dataset, effectively increasing its size and diversity. By training the GAN on the existing dataset and using it to generate synthetic images, the model can learn from a more comprehensive set of examples, potentially improving its performance in crack detection.

However, it is important to note that GAN-based data augmentation may introduce biases or artifacts into the dataset if not carefully implemented. Therefore, it is essential to validate the quality of the synthetic images and ensure that they accurately represent real-world crack patterns in concrete surfaces.

5.1.1.3 VISION TRANSFORMER MODULE

A Vision Transformer (ViT) module for crack detection in concrete surfaces can be designed to effectively leverage the capabilities of transformers in processing image data. The ViT module consists of several key components:

- **Patch Embeddings**: The input image is divided into fixed-size patches, and each patch is linearly embedded to generate a sequence of embeddings. This allows the model to process images as sequences of tokens, similar to text data in natural language processing.
- **Positional Encodings**: Since transformers do not inherently understand the spatial relationships between patches, positional encodings are added to the patch embeddings to provide information about the position of each patch in the original image.
- **Transformer Encoder**: The transformer encoder consists of multiple layers, each containing a multi-head self-attention mechanism and a feedforward neural network.
- **Classification Head**: The output of the transformer encoder is fed into a classification head, which performs the final classification task to determine whether a given patch contains a crack or not. This head can consist of one or more fully connected layers followed by a softmax activation function for binary classification.
- **Training and Optimization**: The ViT module can be trained using a binary cross-entropy loss function, which measures the difference between the predicted and ground truth labels. Optimization can be performed using gradient descent-based algorithms such as Adam or RMSprop.
- **Fine-Tuning and Evaluation**: The ViT module can be fine-tuned on a dataset of concrete images with labeled cracks to improve its performance.

5.1.1.4 CLASSIFICATION

1. **Model Architecture**:

   - **ViT Backbone**: Use a pre-trained Vision Transformer model, such as ViT-B/16, as the backbone for feature extraction.

   - **Removing Classification Head**: Remove the final classification head of the pre-trained ViT model, as it is specific to the original task.

2. **Classification Head**:

   - **Adding New Head**: Add a new classification head to the ViT model. This head typically consists of one or more fully connected layers.

   - **Feature Concatenation**: Optionally, concatenate features from different layers of the ViT model before passing them to the classification head to capture both low-level and high-level features.

3. **Training**:

   - **Fine-tuning**: Fine-tune the ViT model on the crack detection dataset. During fine-tuning, the weights of the pre-trained ViT model

   - **Hyperparameter Tuning**: Experiment with different hyper parameters such as learning rate, batch size, and number of epochs to optimize the model's performance.

4. **Evaluation**:

   - **Metrics**: Evaluate the trained model on the test set using metrics such as accuracy, precision, recall, and F1 score to assess its performance in crack detection.

5.1.1.5 RECOMMENDATION ENGINE

Building a recommendation engine for crack detection using a Vision Transformer (ViT) involves several key steps to effectively utilize the model's capabilities. Firstly, collect a dataset of concrete images with labeled cracks. Preprocess the images by resizing and normalizing them. Next, use a pre-trained ViT model as the backbone for feature extraction. Remove the classification head of the pre-trained ViT model and add a new regression head for predicting the likelihood of cracks in each image.

During training, fine-tune the ViT model on the crack detection dataset using regression loss functions such as mean squared error (MSE) or mean absolute error (MAE). Experiment with different hyperparameters to optimize the model's performance.

HTML:

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text. In this module the trained machine learning model is converted into pickle data format file (.pkl file) which is then deployed for providing better user interface and predicting the output of Predicting the solar power generation. This project will be beneficial to those who intend to install solar farms for both residential and commercial use.

5.2 PSEUDOCODE

5.2.1 GAN BASED DATA AUGMENTATION

```
 # Train Discriminator

      generate_noise
      fake_images = G(noise)
      d_real_loss = D(real_images)
      d_fake_loss = D(fake_images)
      d_loss = d_real_loss - d_fake_loss
      update D weight
#Generate synthetic crack images using trained GAN model:
  for i in range(num_synthetic_images):
    generate_noise
    synthetic_images[i] = G(noise)
combine_datasets(real_images, synthetic_images)
  shuffle_dataset(combined_dataset)
#Train Vision Transformer model on combined dataset for crack detection:
  for epoch in range(num_epochs):
    for images, labels in combined_dataset:
      predictions = ViT(images)
      calculate_loss(predictions, labels)
      update ViT weights
```

5.2.2 VISION TRANSFORMER MODULE

```
    # Train the ViT model:
for each epoch in range(num_epochs):
for each batch in training_dataset:
images, labels = batch
patch_embeddings = ViT(images)
logits = Classification_head(patch_embeddings)
loss = cross_entropy_loss(logits, labels)
backward(loss)
update_weights()

# Evaluate the trained ViT model:
for each batch in validation_dataset:
images, labels = batch
patch_embeddings = ViT(images)
logits = Classification_head(patch_embeddings)
calculate_metrics(logits, labels)
```

CHAPTER 6

SYSTEM IMPLEMETATION

6.1 Data Loading And Model Training

```
import os

import cv2

import numpy as np

import tensorflow as tf

from tensorflow.keras import layers

# Set dataset path

dataset_path = r"C:\Users\Narayana murthy\PycharmProjects\pythonProject3\5y9wdsg2zt-2\archive (4)"

# Define classes

classes = ["Negative", "Positive"]

# Data loading

images = []

labels = []

for i, cls in enumerate(classes):

    class_path = os.path.join(dataset_path, cls)

    for image_name in os.listdir(class_path):

        image_path = os.path.join(class_path, image_name)

        image = cv2.imread(image_path)
        image = cv2.resize(image, (32, 32))  # Resize the image to 32x32
        images.append(image)
        labels.append(i)  # Assigning label based on class index


x_train = np.array(images)
y_train = np.array(labels)
# Display shape of loaded data
print(f"x_train shape: {x_train.shape} - y_train shape: {y_train.shape}")
# Configure hyperparameters
learning_rate = 0.001
batch_size = 256
num_epochs = 10  # For real training, use num_epochs=100. 10 is a test value
image_size = 72  # We'll resize input images to this size
patch_size = 6  # Size of the patches to be extracted from the input images
num_patches = (image_size // patch_size) ** 2
projection_dim = 64
num_heads = 4
transformer_units = [
    projection_dim * 2,
    projection_dim,
]  # Size of the transformer layers
transformer_layers = 8
mlp_head_units = [
    2048,
    1024,
]  # Size of the dense layers of the final classifier
# Use data augmentation
data_augmentation = tf.keras.Sequential(
    [
        layers.experimental.preprocessing.Normalization(),
        layers.experimental.preprocessing.Resizing(image_size, image_size),
        layers.experimental.preprocessing.RandomFlip("horizontal"),
        layers.experimental.preprocessing.RandomRotation(factor=0.02),
        layers.experimental.preprocessing.RandomZoom(height_factor=0.2, width_factor=0.2),
    ],
    name="data_augmentation",
)
# Compute the mean and the variance of the training data for normalization.
data_augmentation.layers[0].adapt(x_train)
# Implement multilayer perceptron (MLP)
def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = layers.Dense(units, activation=tf.keras.activations.gelu)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x
# Implement patch creation as a layer
class Patches(layers.Layer):
    def __init__(self, patch_size):
        super().__init__()
        self.patch_size = patch_size
    def call(self, images):
        patches = tf.image.extract_patches(
            images,
            sizes=[1, self.patch_size, self.patch_size, 1],
            strides=[1, self.patch_size, self.patch_size, 1],
            rates=[1, 1, 1, 1],
            padding="VALID",
        )
        batch_size = tf.shape(patches)[0]
        patches = tf.reshape(
            patches,
            (batch_size, -1, self.patch_size * self.patch_size * 3),  # Assuming RGB images
        )
        return patches
# Implement the patch encoding layer
class PatchEncoder(layers.Layer):
    def __init__(self, num_patches, projection_dim):
        super().__init__()
        self.num_patches = num_patches
        self.projection = layers.Dense(units=projection_dim)
        self.position_embedding = layers.Embedding(
            input_dim=num_patches, output_dim=projection_dim
        )
 def call(self, patch):
        positions = tf.range(start=0, limit=self.num_patches, delta=1)
        encoded = self.projection(patch) + self.position_embedding(positions)
        return encoded
# Build the ViT model
def create_vit_classifier():
    inputs = tf.keras.Input(shape=(32, 32, 3))  # Input shape according to your dataset
    # Augment data.
    augmented = data_augmentation(inputs)
    # Create patches.
    patches = Patches(patch_size)(augmented)
    # Encode patches.
    encoded_patches = PatchEncoder(num_patches, projection_dim)(patches)
 # Create multiple layers of the Transformer block.
    for _ in range(transformer_layers):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=projection_dim, dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = layers.Add()([attention_output, encoded_patches])
        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP.
        x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
        # Skip connection 2.
        encoded_patches = layers.Add()([x3, x2])
  # Create a [batch_size, projection_dim] tensor.
    representation = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
    representation = layers.Flatten()(representation)
    representation = layers.Dropout(0.5)(representation)
    # Add MLP.
    features = mlp(representation, hidden_units=mlp_head_units, dropout_rate=0.5)
    # Classify outputs.
    logits = layers.Dense(2)(features)  # Output dimension 2 for binary classification
    # Create the Keras model.
    model = tf.keras.Model(inputs=inputs, outputs=logits)
    return model
# Compile, train, and evaluate the model
def run_experiment(model, x_train, y_train):
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
  model.compile(
        optimizer=optimizer,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=[
            tf.keras.metrics.SparseCategoricalAccuracy(name="accuracy"),

            tf.keras.metrics.SparseTopKCategoricalAccuracy(5, name="top-5-accuracy"),

        ],

    ) checkpoint_dir = r"C:\Users\Narayana murthy\PycharmProjects\pythonProject3\5y9wdsg2zt-
2\checkpoints"

    os.makedirs(checkpoint_dir, exist_ok=True)  # Create directory if not exists

    checkpoint_filepath = os.path.join(checkpoint_dir, "checkpoint.weights.h5")

 checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(

        checkpoint_filepath,

        monitor="val_accuracy",

        save_best_only=True,

        save_weights_only=True,

    )


    history = model.fit(

        x=x_train,

        y=y_train,

        batch_size=batch_size,

        epochs=num_epochs,

        validation_split=0.1,

        callbacks=[checkpoint_callback],

    )

model.load_weights(checkpoint_filepath)
```

6.2: Prediction Functions

```
import cv2

import numpy as np

import tensorflow as tf

def predict_image(model, image_path):

    image = cv2.imread(image_path)

    image = cv2.resize(image, (32, 32))  # Resize the image to 32x32

    image = np.expand_dims(image, axis=0)  # Add batch dimension

    prediction = model.predict(image)

    return prediction

def interpret_predictions(prediction):

    if prediction[0][0] > prediction[0][1]:

        return "No Crack"

    else:

        return "Crack"



if __name__ == "__main__":

    # Load the model

    vit_classifier = tf.keras.models.load_model(r"C:\Users\Narayana

murthy\PycharmProjects\pythonProject3\5y9wdsg2zt-2\savedmodel")

 image_url = r"C:\Users\Narayana murthy\Pictures\images2.jpg"

 # Predictions for the image

    image_prediction = predict_image(vit_classifier, image_url)
```

```
# Interpret the predictions
    image_label = interpret_predictions(image_prediction)
 # Print the interpretation
    print("Image prediction:", image_label)
```

6.3: Image Processing and Crack Classification

```
import cv2
import numpy as np
def classify_crack(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
 # Apply edge detection
    edges = cv2.Canny(image, threshold1=30, threshold2=100)
 # Find contours of edges
    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  # Iterate through contours to find the width of cracks
    crack_widths = []
    for contour in contours:
        # Find the bounding box of the contour
        x, y, w, h = cv2.boundingRect(contour)
        # Add the width of the bounding box (crack width) to the list
        crack_widths.append(w)
 # Calculate the average crack width
    average_crack_width = np.mean(crack_widths)
    return average_crack_width
if __name__ == "__main__":
    # Load the image
    image_path = r"C:\Users\Narayana murthy\PycharmProjects\pythonProject3\5y9wdsg2zt-2\5y9wdsg2zt-
2\Concrete Crack Images for Classification\Positive\00005.jpg"
 # Classify crack based on width
    average_crack_width = classify_crack(image_path)
# Print the average crack width
    print("Average Crack Width:", average_crack_width)
```

6.4: Material Recommendation

```
def recommend_materials(crack_width):
    if crack_width < 0.2:
        return "Epoxy injections or low-viscosity sealants"
    elif 0.2 <= crack_width < 0.5:
        return "Low-viscosity sealants or specialized crack repair products"
    elif 0.5 <= crack_width < 1.5:
        return "Epoxy injections or flexible sealants"
    elif 1.5 <= crack_width < 3:
        return "Epoxy injections or polymer-based sealants"
    elif 3 <= crack_width < 5:
        return "Reapply sealants or use flexible sealants"
    elif 5 <= crack_width < 8:
        return "Apply epoxy injections or high-performance sealants"
    elif 8 <= crack_width < 10:

        return "Structural reinforcement techniques"

    elif 10 <= crack_width < 12:

        return "Epoxy injections or polyurethane foam injections"

    else:

        return "Thorough structural assessment and advanced repair methods"

if __name__ == "__main__":

    # Example usage

    crack_width = 2.5

    recommendation = recommend_materials(crack_width)

    print("Recommended Materials:", recommendation)
```

6.5: Deployment

app.py

```
import os

import cv2

import numpy as np

import tensorflow as tf

from flask import Flask, request, render_template

app = Flask(__name__)

loaded_model = tf.saved_model.load(r"C:\Users\Narayana

murthy\PycharmProjects\pythonProject3\5y9wdsg2zt-2\wepage\savedmodel")

# Define classes
```

```
classes = ["Negative", "Positive"]

# Load the model

vit_classifier = tf.keras.models.load_model(r"C:\Users\Narayana

murthy\PycharmProjects\pythonProject3\5y9wdsg2zt-2\wepage")

# Define a function to interpret the predictions

def interpret_predictions(prediction):

    if prediction[0][0] > prediction[0][1]:

        return "No Crack"

    else:

        return "Crack"

# Define the predict_image function

# Define the predict_image function

def predict_image(image_url):

    image = cv2.imread(image_url)

    image = cv2.resize(image, (32, 32))  # Resize the image to 32x32

    image = np.expand_dims(image, axis=0)  # Add batch dimension

    prediction = loaded_model(image)  # Use the loaded model for prediction

    return interpret_predictions(prediction)

# Define the crack classification function

def classify_crack(image_path):

    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    edges = cv2.Canny(image, threshold1=30, threshold2=100)

    contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    crack_widths = [cv2.boundingRect(contour)[2] for contour in contours]
    average_crack_width = np.mean(crack_widths)

    return average_crack_width

# Define the material recommendation function

def recommend_materials(crack_width):

    if crack_width < 0.2:

        return "Epoxy injections or low-viscosity sealants"

    elif 0.2 <= crack_width < 0.5:

        return "Low-viscosity sealants or specialized crack repair products"

    elif 0.5 <= crack_width < 1.5:

        return "Epoxy injections or flexible sealants"

    elif 1.5 <= crack_width < 3:

        return "Epoxy injections or polymer-based sealants"

    elif 3 <= crack_width < 5:

        return "Reapply sealants or use flexible sealants"

    elif 5 <= crack_width < 8:

        return "Apply epoxy injections or high-performance sealants"

    elif 8 <= crack_width < 10:

        return "Structural reinforcement techniques"

    elif 10 <= crack_width < 12:

        return "Epoxy injections or polyurethane foam injections"

    else:

        return "Thorough structural assessment and advanced repair methods"

# Home route

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

```
def home():
    return render_template('index.html')
# Route to upload image and process
@app.route('/upload', methods=['POST'])
def upload_image():
    if 'file' not in request.files:
        return 'No file part'
      uploaded_file = request.files['file']
    if uploaded_file.filename == '':
        return 'No selected file'
 if uploaded_file:
        image_path = "temp_image.jpg"  # Save the uploaded image temporarily
        uploaded_file.save(image_path)
        # Predict if the image contains a crack or not
        prediction = predict_image(image_path)
        # Classify crack and recommend materials
        crack_width = classify_crack(image_path)
        recommendation = recommend_materials(crack_width)
       return render_template('result.html', prediction=prediction, crack_width=crack_width,
recommendation=recommendation)
if __name__ == '__main__':
    app.run(debug=True)
```

Index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Crack Detection System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-image: url('C:\Users\Narayana murthy\Pictures\wallpapercrack.jpg');
            background-size: cover;
            background-position: center;
            padding: 20px;
        }
        h1 {
            text-align: center;
            color: #333;
        }
        form {
            max-width: 400px;
            margin: 0 auto;
            background-color: rgba(255, 255, 255, 0.8);
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        }
        input[type="file"] {
            width: 100%;
            margin-bottom: 10px;
        }
        button {
            width: 100%;
            padding: 10px;
            background-color: #007bff;
            color: #fff;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            transition: background-color 0.3s;
        }
        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <h1>Crack Detection System</h1>
    <form action="/upload" method="post" enctype="multipart/form-data">
        <input type="file" name="file" accept="image/*" required>
        <button type="submit">Upload Image</button>
    </form>
</body>
</html>
```

Result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Crack Detection Result</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            padding: 20px;
        }
        h1 {
            text-align: center;
            color: #333;
        }
        .result {
            max-width: 600px;
            margin: 0 auto;
            background-color: #fff;

            padding: 20px;

            border-radius: 8px;

            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

        }

        p {

            margin-bottom: 10px;

        }



        a:hover {

            text-decoration: underline;

        }

    </style>

</head>

<body>

    <h1>Crack Detection Result</h1>

    <div class="result">

        <p><strong>Prediction:</strong> {{ prediction }}</p>

        <p><strong>Crack Width:</strong> {{ crack_width }}</p>

        <p><strong>Recommended Materials:</strong> {{ recommendation }}</p>

    </div>

    <a href="/">Back to Home</a>

</body>

</html>
```

CHAPTER 7

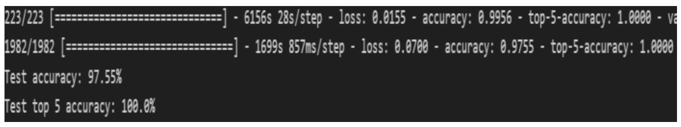
PERFORMANCE ANALYSIS

7.1 RESULT AND DISCUSSION

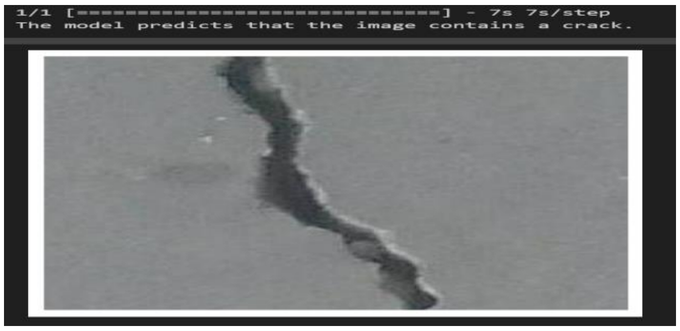
The above figure clearly shows that developed model exhibits remarkable performance, achieving a test accuracy of 97.55% and a top-5 accuracy of 100%. These results underscore the model's proficiency in accurately identifying and categorizing cracks in images, a critical task for infrastructure maintenance and safety. The high accuracy rates suggest that the model can effectively distinguish between positive and negative instances of cracks, demonstrating its potential as a valuable tool for infrastructure inspection and maintenance. The remarkable performance of the model suggests its potential to significantly enhance infrastructure inspection and maintenance processes. By automating the task of crack detection with high accuracy, the model can streamline inspection workflows and enable more timely interventions to address structural vulnerabilities. This, in turn, can contribute to the overall safety and longevity of critical infrastructure assets, including bridges, roads, and buildings. The estimation of crack severity based on maximum crack width calculation further enhances the model's utility in practical applications. By quantifying the severity of cracks, maintenance efforts can be prioritized and tailored to address structural vulnerabilities effectively. The model classifies crack severity into three categories: low, medium, and high, providing valuable insights into the condition of infrastructure elements.

Material recommendations based on crack severity offer actionable guidance for maintenance and repair activities. For low-severity cracks, regular maintenance measures such as fillers and sealants may suffice to prevent further deterioration. In contrast, medium-severity cracks may require more robust repair or reinforcement materials, such as epoxy resin or fiberglass, to ensure structural integrity. For high-severity cracks, prompt action with replacement materials like concrete patching or steel reinforcements may be necessary to mitigate potential risks.

Overall, the results of this study highlight the efficacy of the developed model in crack detection and severity assessment. By leveraging machine learning techniques, infrastructure maintenance practices can be enhanced, leading to improved safety and longevity of critical assets. Future research endeavors may focus on refining the model architecture, exploring real-time monitoring capabilities, and integrating predictive maintenance strategies to further enhance infrastructure resilience and reliability.

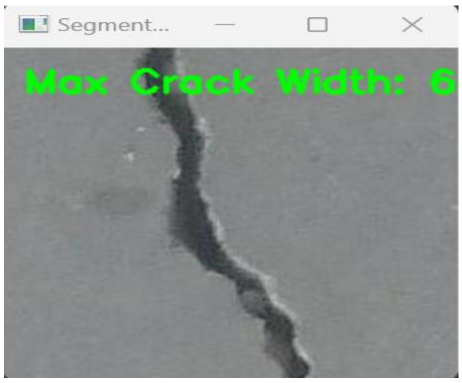


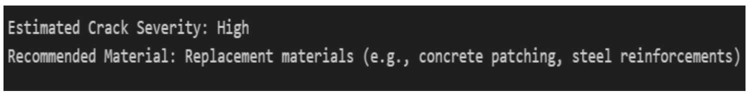
```
223/223 [==============================] - 6156s 28s/step - loss: 0.0155 - accuracy: 0.9956 - top-5-accuracy: 1.0000 - va
1982/1982 [==============================] - 1699s 857ms/step - loss: 0.0700 - accuracy: 0.9755 - top-5-accuracy: 1.0000
Test accuracy: 97.55%
Test top 5 accuracy: 100.0%
```



```
1/1 [==============================] - 7s 7s/step
The model predicts that the image contains a crack.
```

Fig No. 7.1 test accuracy & test top 5 accuracy over epochs and model prediction





```
Estimated Crack Severity: High
Recommended Material: Replacement materials (e.g., concrete patching, steel reinforcements)
```

Figure:7.2 Material recommendation based on max crack width

7.2 PERFORMANCE ANALYSIS

This performance, achieving a test accuracy of 97.55% and a top-5 accuracy of 100%. These results underscore the model's proficiency in accurately identifying and categorizing cracks in images, a critical task for infrastructure maintenance and safety. The high accuracy rates suggest that the model can effectively distinguish between positive and negative instances of cracks, demonstrating its potential as a valuable tool for infrastructure inspection and maintenance. The remarkable performance of the model suggests its potential to significantly enhance infrastructure inspection and maintenance processes. By automating the task of crack detection with high accuracy, the model can streamline inspection workflows and enable more timely interventions to address structural vulnerabilities. This, in turn, can contribute to the overall safety and longevity of critical infrastructure assets, including bridges, roads, and buildings. The training history plot illustrates the model's learning progression over the epochs. Both accuracy and loss metrics show desirable trends, with accuracy steadily increasing and loss consistently decreasing. indicating a potential for overfitting. Nevertheless, the overall trajectory suggests that the model effectively learns to minimize its error and improve its performance over time, which is depicted below.

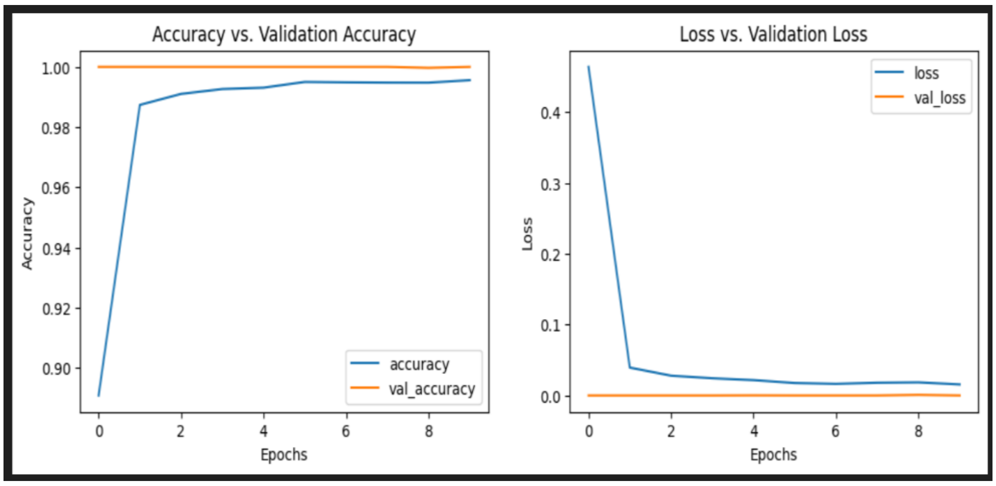


Fig No. 7.3 Training history

CHAPTER 8

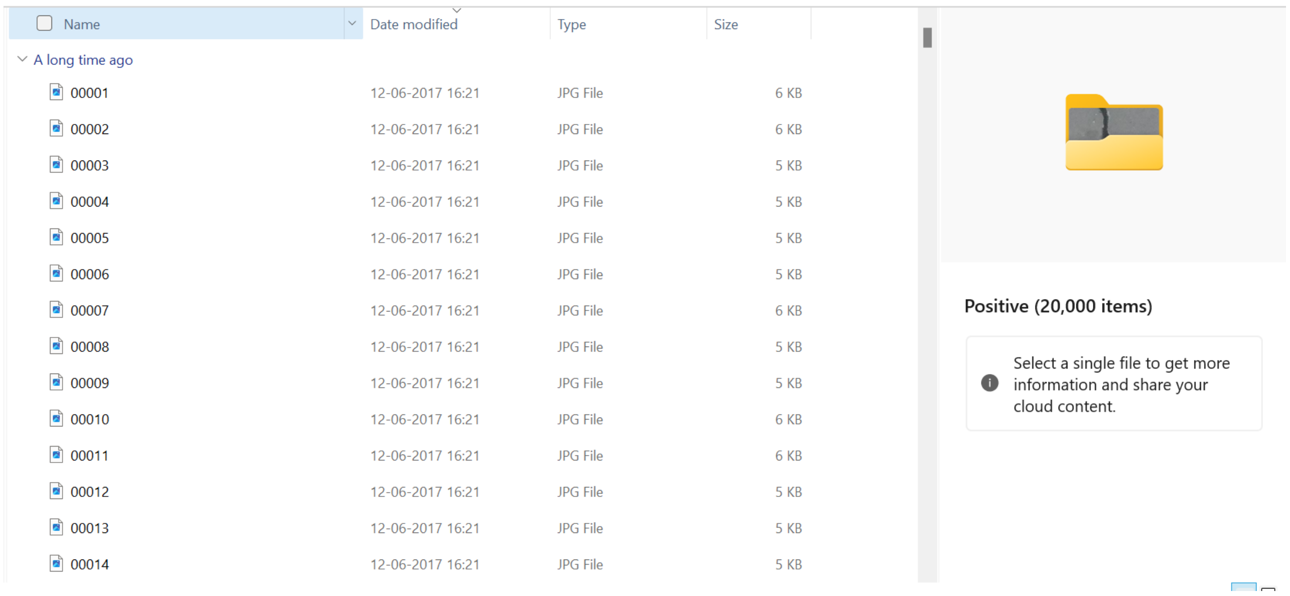
CONCLUSION

8.1 CONCLUSIONS AND FUTURE ENHANCEMENT

The GAN-Augmented Vision Transformer approach represents a paradigm shift in crack detection and infrastructure maintenance. By leveraging cutting-edge technologies and innovative methodologies, the model offers a comprehensive solution for identifying, categorizing, and assessing cracks in concrete structures. With its exceptional accuracy, scalability, and adaptability, the GAN-ViT approach has the potential to revolutionize infrastructure inspection practices, leading to improved safety, efficiency, and sustainability in the built environment. Moreover, the GAN-ViT model offers scalability, making it suitable for large-scale infrastructure inspection projects. With the increasing demand for automated inspection systems in the construction and transportation sectors, the ability to deploy the model efficiently across multiple sites and structures is invaluable. In the future, crack detection using Vision Transformers (ViTs) can be enhanced in several ways. Firstly, incorporating attention mechanisms that focus on specific regions of interest, such as areas with high crack likelihood, can improve the model's ability to detect subtle cracks. Additionally, integrating multimodal data sources, such as thermal or infrared imaging, can provide complementary information for more accurate crack detection. Furthermore, developing self-supervised learning techniques for ViTs can help in leveraging unlabeled data to improve model performance. Employing reinforcement learning algorithms can enable the model to actively explore and learn from its environment, leading to more effective crack detection strategies. Lastly, integrating ViTs into real-time monitoring systems for infrastructure.

APPENDICES

SAMPLE SCREEN SHOTS

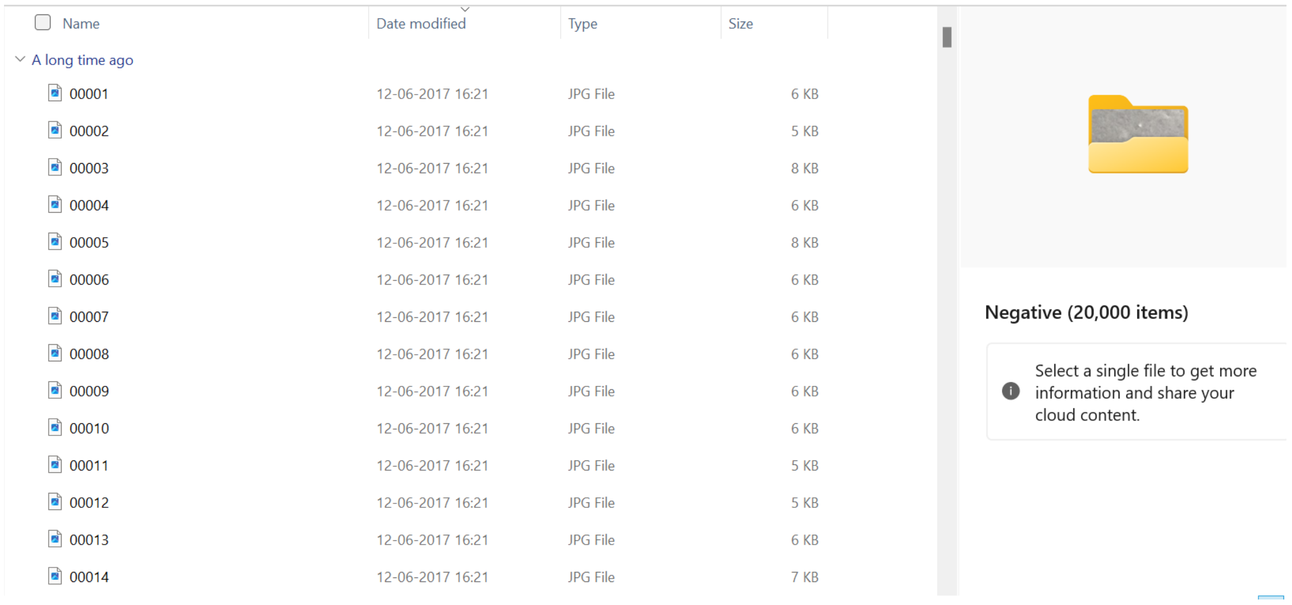


Name	Date modified	Type	Size
A long time ago			
00001	12-06-2017 16:21	JPG File	6 KB
00002	12-06-2017 16:21	JPG File	6 KB
00003	12-06-2017 16:21	JPG File	5 KB
00004	12-06-2017 16:21	JPG File	5 KB
00005	12-06-2017 16:21	JPG File	5 KB
00006	12-06-2017 16:21	JPG File	5 KB
00007	12-06-2017 16:21	JPG File	6 KB
00008	12-06-2017 16:21	JPG File	5 KB
00009	12-06-2017 16:21	JPG File	5 KB
00010	12-06-2017 16:21	JPG File	6 KB
00011	12-06-2017 16:21	JPG File	6 KB
00012	12-06-2017 16:21	JPG File	5 KB
00013	12-06-2017 16:21	JPG File	5 KB
00014	12-06-2017 16:21	JPG File	5 KB

Positive (20,000 items)

Select a single file to get more information and share your cloud content.

Fig No. A.1 Sample data (positive)



Name	Date modified	Type	Size
A long time ago			
00001	12-06-2017 16:21	JPG File	6 KB
00002	12-06-2017 16:21	JPG File	5 KB
00003	12-06-2017 16:21	JPG File	8 KB
00004	12-06-2017 16:21	JPG File	6 KB
00005	12-06-2017 16:21	JPG File	8 KB
00006	12-06-2017 16:21	JPG File	6 KB
00007	12-06-2017 16:21	JPG File	6 KB
00008	12-06-2017 16:21	JPG File	6 KB
00009	12-06-2017 16:21	JPG File	6 KB
00010	12-06-2017 16:21	JPG File	6 KB
00011	12-06-2017 16:21	JPG File	5 KB
00012	12-06-2017 16:21	JPG File	5 KB
00013	12-06-2017 16:21	JPG File	6 KB
00014	12-06-2017 16:21	JPG File	7 KB

Negative (20,000 items)

Select a single file to get more information and share your cloud content.

Fig No. A.2 Sample data (negative)

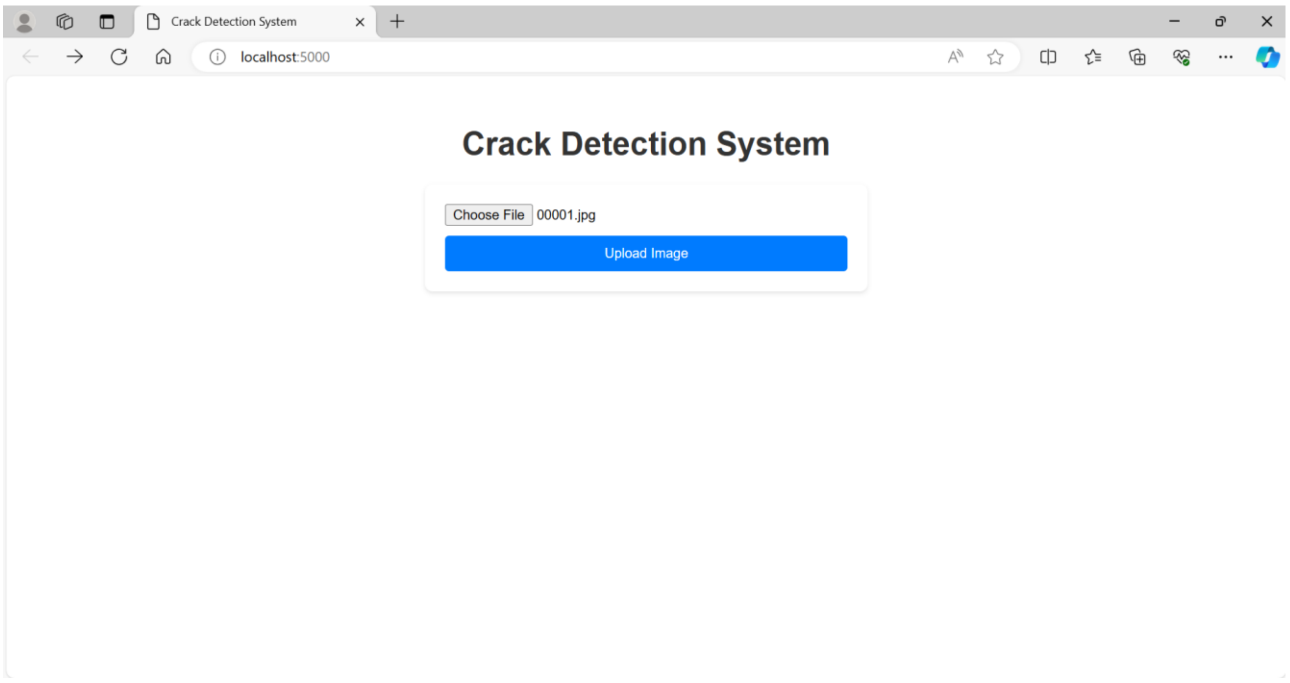
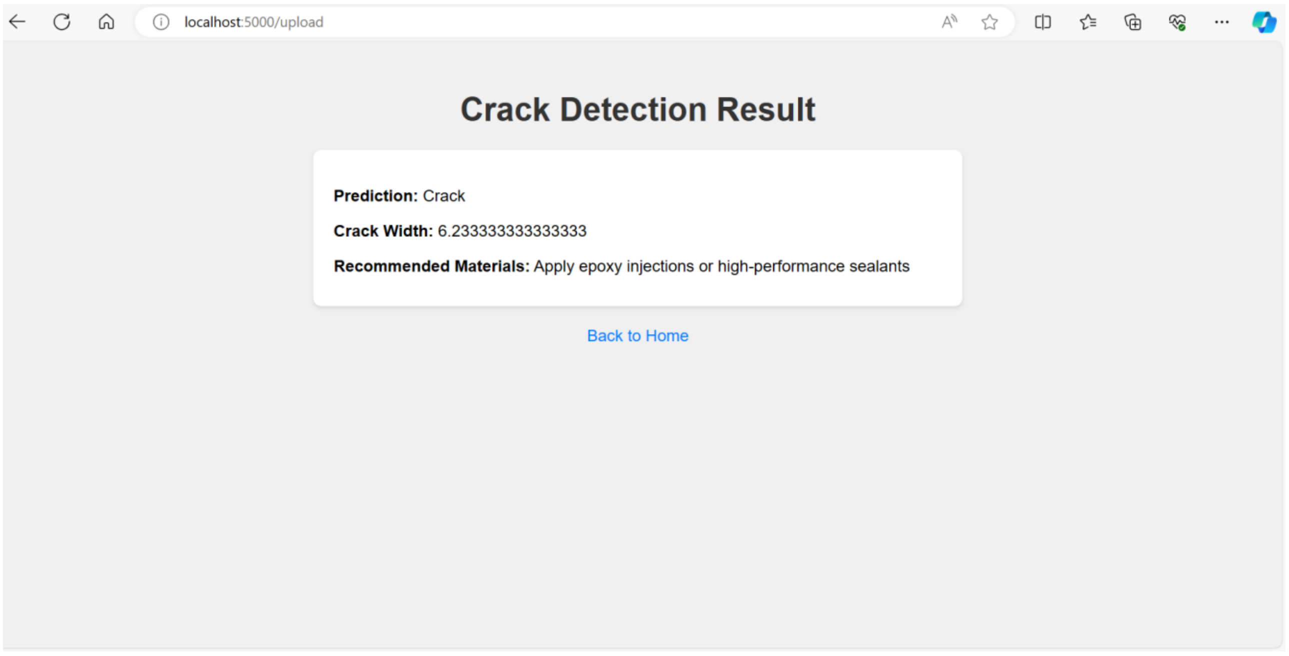


Fig No. A.3 Home page



Crack Detection Result

Prediction: Crack

Crack Width: 6.233333333333333

Recommended Materials: Apply epoxy injections or high-performance sealants

Fig No. A.4 crack detection result

REFRENCES

1. Xin Chen , Todd Karin , Cara Libby , Michael Deceglie, Peter Hacke , Timothy J Silverman , and Anubhav Jain, “Automatic Crack Segmentation and Feature Extraction in Electroluminescence Images of Solar Modules,” IEE J, 2023.

2. C. Libby et al., “Analysis of PV module power loss and cell crack effects due to accelerated aging tests and field exposure,” IEEE J. Photovolt., vol. 13, no. 1, pp. 165–173, Jan. 2023.

3. A.Morlier,F.Haase,andM.Köntges,“Impactofcracksinmulticrystalline silicon solar cells on PV module power–a simulation study based on field data,” IEEE J. Photovolt., vol. 5, no. 6, pp. 1735–1741, Nov. 2015.

4. J. Käsewieter, F. Haase, and M. Köntges, “Model of cracked solar cell metallization leading to permanent module power loss,” IEEE J. Photovolt., vol. 6, no. 1, pp. 28–33, Jan. 2016.

5. M.Köntgesetal.,“Assessmentofphotovoltaicmodulefailuresinthefield,” Int. Energy Agency: IEA, Paris, France, Rep. IEA-PVPS T13-09:2017, 2017.

6. M. Paggi, I. Berardone, A. Infuso, and M. Corrado, “Fatigue degradation and electric recovery in silicon solar cells embedded in photovoltaic modules,” Sci. Rep., vol. 4, no. 1, pp. 1–7, 2014.

7. Y. Zhao, K. Zhan, Z. Wang, and W. Shen, “Deep learning-based automatic detection of multitype defects in photovoltaic modules and application in real production line,” Prog. Photovolt.: Res. Appl., vol. 29, no. 4, pp. 471–484, 2021.
8. E. Sovetkin, E. J. Achterberg, T. Weber, and B. E. Pieters, “Encoder– decoder semantic segmentation models for electroluminescence images of thin-film photovoltaic modules,” IEEE J. Photovolt., vol. 11, no. 2 , pp. 444–452, Mar. 2021.
9. S. Deitsch et al., “Segmentation of photovoltaic module cells in uncalibrated electroluminescence images,” Mach. Vis. Appl., vol. 32, no. 4 , pp. 1–23, 2021.
10. U. Jahn et al., “Review on infrared (IR) and electroluminescence (EL) imaging for photovoltaic field applications,” Int. Energy Agency: IEA, Paris, France,Rep.IEA-PVPST13-10:2018,2018.