A Deep learning and GAI fusion for building structure crack detection

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO THE PROJECT

The identification and timely detection of structural cracks are imperative for ensuring the safety and longevity of critical infrastructure. In response to these challenges, the study introduces an innovative approach that harnesses the power of deep learning, specifically Convolutional Neural Networks (CNNs), to automate and enhance the crack identification process.

The core methodology of this study involves training the CNN on a diverse dataset meticulously curated to encompass various crack types and sizes. This approach ensures that the model becomes proficient in recognizing intricate patterns that distinguish between cracked and non-cracked surfaces. The inclusion of diverse crack scenarios in the training set enables the model to develop a nuanced understanding of the wide spectrum of crack formations encountered in real-world infrastructure.

The study employs transfer learning techniques to refine the model's ability to generalize across different scenarios. Transfer learning leverages pre-existing knowledge from the initial training phases, enabling the model to adapt swiftly to new and unseen crack patterns. This transferability enhances the model's efficiency and accuracy in identifying structural cracks in diverse environments and materials.

By automating the identification process and enhancing the accuracy of detection, the proposed approach offers a more efficient and scalable solution to address the challenges associated with ensuring infrastructure integrity. The incorporation of deep learning, particularly CNNs, into crack detection processes marks a significant step forward in the quest for safer and more resilient infrastructure. This methodology offers a paradigm shift in crack detection, providing a timely, accurate, and scalable solution that can significantly contribute to ensuring the structural integrity of our vital infrastructure systems.

1.2 STATEMENT OF THE PROBLEM

The problem statement is "To develop an Automated System to classify types of cracks using the Deep Learning Techniques."

1.3 SYSTEM SPECIFICATIONS

Table 1: Table Specifications

Processor	9th Gen Intel Core i5-9300H
Operating System	Windows 11 Home
RAM	8 GB 2.40 GHz

Table 2: Hardware Interface

Processor	GPU
Storage	Google Drive

Table 3: Software Interface

IDE	Google Colab
Programming Language	Python

CHAPTER 2

LITERATURE REVIEW

The paper employs Machine Learning algorithms, including a Convolutional Neural Network (CNN) powered by TensorFlow, combined with image processing techniques such as the Sobel Edge Detection Algorithm. Utilizing drones for real-time data, it achieves an impressive precision of 86%, recall of 96%, and an F1 score of 91% in detecting cracks on the outer parts of high-rise buildings. The use of Sobel edge detection enhances the system's efficiency in edge identification, providing a fundamental and widely adopted method for crack detection. [1]

Focusing on the health monitoring of old buildings in Bangladesh, the study employs non-destructive testing (NDT) and utilizes a dataset of 500 cellphone camera photographs for detecting cracks and dampness through YOLOv4-Tiny, a deep learning-based Object Detection model. With a precision of 72.46% and recall of 75%, the implementation of YOLOv4 enhances accuracy in detecting structural vulnerabilities. [2]

The study introduces an autonomous image-based method, employing CNN-SVM, for the precise detection of cracks on historic brick walls. With an accuracy of 85.94%, superior precision, recall, and F1 score compared to CNN, the model refinement includes adding more layers and adjusting settings to enhance efficiency. [3]

The study employs Machine Learning and Thermography to accurately identify cracks in buildings, achieving an impressive accuracy of 96.83%, precision of 96.91%, recall of 96.73%, and F1 score of 96.83%. The recommendation emphasizes considering Convolutional Neural Networks (CNNs) as the primary choice for the analysis of thermal imagery in crack detection. [4]

The system utilizes Microwave Imaging and advanced techniques, including Artificial Neural Networks (ANN) and Bayesian Classifiers, for automatic crack detection in columns. With a low Crack Error Rate (CER) of 0.1, the study emphasizes that ANNs are a suitable choice for building effective automatic crack recognition systems, contributing to enhanced structural assessment. [5]

The paper introduces a Modified Deep CNN Model (MDCNN) to enhance crack and damage identification in civil infrastructures, achieving an accuracy of 98.2%, recall of 87.1%, and an F1 score of 85.01% using 1300 photos of cracks. The study suggests that a CNN design with SVM integration emerges as the recommended algorithm for this application, surpassing limitations in automated crack detection.[6]

The paper introduces a novel method for concrete crack detection, utilizing drones to capture images of exterior defects. Employing Deep Convolutional Neural Networks (DCNN), DDS,

DIP, ResNet50, and VGG16, the system achieves an impressive accuracy of 98.4658% on the Concrete Crack Dataset. Notably, ResNet50 and VGG16 exhibit superior performance, making them effective choices for concrete crack detection in this setup.[7]

The study presents an image processing approach for assessing wall surfaces in high-rise buildings, utilizing Machine Learning and Steerable Filters. With a dataset of 500 samples, the implemented Support Vector Machine (SVM) achieves an accuracy of 85.33%, outperforming traditional methods, highlighting the potential of this automatic approach for periodic surveys of concrete wall structures. The focus on feature extraction and pattern classification enhances the efficiency of wall defect recognition. [8]

The research employs a machine learning approach, utilizing a Convolutional Neural Network (CNN) with VGG16-Net architecture, for detecting cracks in concrete structures. Image segmentation using the gradient boosting algorithm is applied to datasets from Kaggle, resulting in the VGG16-Net model achieving impressive precision (98.0%), accuracy (98.0%), f1-score (98.50%), and recall (99.10%). Additionally, the ViT model demonstrates competitive performance, with an overall validation accuracy of 96%, showcasing the effectiveness of both models in crack detection. [9]

The paper introduces a nondestructive method for predicting concrete crack depth using thermal images and machine learning, achieving accuracies of 76.74% and 80.99%. Employing innovative techniques like PCA, SVD, and ICA, the study demonstrates the potential for practical applications in non-destructive testing and structural engineering, making it a valuable resource for researchers and practitioners in the field. [10]

The study employs Transfer Learning with four pre-trained CNN models (VGG16, ResNet18, DenseNet161, and AlexNet) for concrete crack detection, where AlexNet achieves remarkable testing accuracy (99.90%), precision (99.92%), recall (99.80%), and F1-score (99.86%) on the CCIC dataset. The paper suggests future research directions to enhance the model's robustness and adaptability in concrete crack detection. [11]

The study introduces a deep hierarchical CNN architecture for crack detection and damage assessment in civil infrastructures using UAV-collected images, achieving an accuracy of 93.9%, precision of 83.8%, recall of 87.9%, and an F-Score of 85.81%. The paper emphasizes the significance of AI and ML techniques, particularly CNN-based models, for automatic feature learning and effective crack detection in structural inspections.[12]

The paper proposes a concrete crack detection system using a combined Convolutional Neural Network (CNN) and Support Vector Machine (SVM) model, achieving an accuracy of 90.76%. The study suggests the potential for further efficiency improvement through fine-tuning CNN parameters, including the size of convolution filters and max-pooling layers.[13]

The study presents an innovative approach for automated defect detection and classification in ashlar masonry walls using machine learning, achieving a recall of 0.9 and precision of 0.42 for geometry-related defects. The research employs Terrestrial Laser Scanning (TLS) and photogrammetry for data acquisition, followed by segmentation of the obtained point cloud data into individual stone units.[14]

The study evaluates the performance of pretrained models, including ResNet50, VGG16, and VGG19, in surface crack detection, with ResNet50 achieving a notable 99.17% training accuracy. The research suggests further exploration of crack segmentation based on different CNN models, emphasizing the effectiveness of ResNet50 and VGG19 in crack detection.[15]

The study employs a VGG-16 network, RPN network, and Faster R-CNN detection networks for crack detection in green concrete, achieving an average detection precision of 98.29% at a speed of 12.26fps. The research highlights the improvement in accuracy and speed by fusing features from different layers of the VGG-16 network, resulting in a 95.92% detection accuracy and 22.45fps.[16]

The research develops a Convolutional Neural Network (CNN) for surface crack detection, achieving an impressive accuracy of 97.52% and a low loss of 8.14%. The study emphasizes the model's efficiency, suggesting its deployment for real-time applications on websites due to its high accuracy in distinguishing between cracked and normal surfaces.[17]

The study introduces a shallow CNN architecture, OLeNet, for surface concrete crack detection, achieving an impressive accuracy of 99.8% with minimal computation, outperforming deep CNN architectures. The research emphasizes the efficiency of this approach, providing accurate monitoring of structural damage using low-power computational devices.[18]

The study employs ground-penetrating radar (GPR) and YOLO deep learning models for detecting concealed cracks, achieving a precision of 0.71, recall of 0.84, and F1-score of 0.78. Published in 2021, the research highlights the effectiveness of the YOLOv4-tiny model, operating at 10.16 frames per second and outperforming YOLOv3 models in mean Average Precision (mAP) values, contributing to concealed crack detection using 3-D GPR.[19]

The paper introduces a two-step deep learning method for automated façade crack detection, employing a CNN for classification and a U-Net model for segmentation, achieving precision of 97.40%, recall of 90.05%, and an F1-score of 93.49. The research suggests future studies on computing algorithms for automated assessment of detected cracks in UAV-captured façade images.[20]

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Cracks in structures or materials can compromise their integrity, leading to potential safety hazards and reduced performance. Understanding and addressing cracks promptly is crucial for maintaining structural integrity and preventing catastrophic failures. The manual identification of cracks are as follows:

Visual Inspection

- Thorough Examination:

  Conduct a visual inspection of the material or structure to identify visible cracks. Look for discontinuities, changes in surface color, or irregular patterns.

- Surface Preparation:

  Clean the surface to remove any debris, coatings, or substances that may obstruct crack visibility.

- Lighting Conditions:

  Use appropriate lighting, such as side or oblique lighting, to enhance the visibility of cracks. Varying angles of light can reveal surface irregularities.

- Magnification:

  Employ magnifying tools, like magnifying glasses or microscopes, for a closer examination, especially for finer cracks that may be less visible to the naked eye.

Measurement and Documentation

- Crack Measurement:

  Measure the length, width, and depth of identified cracks using precise tools. Accurate measurements help in assessing the severity of the cracks.

- Photographic Documentation:

  Capture clear photographs of the cracks from different angles to document their location and characteristics.

Assessing Severity of Cracks

- Width and Length:

  Evaluate the dimensions of cracks. Generally, wider and longer cracks may indicate more significant structural issues.

- Depth:

  Assess the depth of the cracks to understand their penetration into the material.

- Location:

  Consider the location of cracks concerning critical load-bearing areas or stress points.

Structural Analysis

- Impact on Integrity:

  Perform a structural analysis to understand how cracks may affect the overall integrity of the material or structure.

- Propagation:

  Investigate if cracks show signs of propagation or if they are stable. This helps predict potential future issues.

Importance of Severity Assessment

- Risk Mitigation:

  Understanding the severity of cracks enables prioritization of repairs, reducing the risk of sudden failures.

- Maintenance Planning:

  Severity assessment guides maintenance planning by identifying critical areas that require immediate attention.

The manual process is time taking and is not cost effective. Since this process is done by human there is more probability that an error occurs. In manual identification and severity assessment of cracks are vital steps in ensuring the safety and longevity of structures and materials. Thorough visual inspection, precise measurement, and a comprehensive assessment process help in making informed decisions for maintenance and repairs. Regular monitoring and adherence to safety protocols are essential to prevent further deterioration and ensure the continued reliability of the inspected materials or structures.

3.2 LIMITATIONS OF THE EXISTING SYSTEM

The existing systems for the identification of cracks, whether manual or automated, may encounter several limitations. These limitations can impact the accuracy, efficiency, and reliability of crack detection methods.

- Dependence on Surface Visibility:

  Many crack identification methods rely on visual inspection, which can be hindered by poor lighting conditions, surface roughness, or the presence of coatings and contaminants.

- Limited to Surface Cracks:

  Some techniques, especially visual inspection and certain non-destructive testing methods, may only detect surface cracks. Internal or subsurface cracks may go unnoticed.

- Material Dependency:

  Different materials have different properties, and some crack detection methods may be more effective on certain materials than others. The existing system may not be universally applicable.

- Limited Sensitivity:

  The sensitivity of some non-destructive testing methods may not be sufficient to detect small or early-stage cracks, potentially leading to undetected issues.

- Complex Structures:

  Identifying cracks in complex structures with intricate geometries or overlapping components can be challenging. The existing system may struggle to navigate and inspect such configurations effectively.

- Time-Consuming Manual Processes:

  Manual crack identification methods can be time-consuming, especially for large structures or areas. This can lead to delays in inspections and potential risks of overlooking critical cracks.

- Cost of Equipment and Training:

  Certain advanced non-destructive testing methods require expensive equipment and specialized training. This can limit their accessibility, particularly for smaller organizations or projects with budget constraints.

- Difficulty in Detecting Microcracks:

Microcracks, which are extremely small and often invisible to the naked eye, may be challenging to detect using conventional methods.

- Human Subjectivity:

  Visual inspection methods may be subjective and dependent on the experience and expertise of the inspector, leading to variability in results.

3.3 PROPOSED SYSTEM

We collected the images of 4 different types of cracks from various sources and saved it in different folders. The dataset was divided into learning dataset and test dataset. The training dataset is split into training set and validation set. Since we used deep learning technique the extraction of features is automatic. To train the model we used the below algorithms and selected the best model based on the accuracy and evaluation metrics of each algorithm. for the best models we gained the confusion matrix and the classification report

- CNN model
- DenseNet_121 Model
- EfficientnetB7 Model
- iv) MobileNet_V3 Model
- VGG_16 Model
- ResNet_152 Model

Crack Craft proposes an innovative fusion of Deep Learning and Generative Adversarial Intelligence (GAI) for the detection of cracks in building structures. Leveraging the power of Convolutional Neural Networks (CNNs) within the realm of Deep Learning, the model aims to efficiently learn intricate patterns associated with crack formations. Additionally, the integration of GAI introduces a novel dimension by generating synthetic images of cracks, addressing data limitations and enhancing the model's adaptability. This synergistic fusion not only aims for superior accuracy in crack detection but also sets the stage for a more robust and versatile approach to structural health monitoring, showcasing the potential of advanced technologies in revolutionizing the field of infrastructure integrity.

3.4 ADVANTAGES OF THE PROPOSED SYSTEM

- Surface Visibility Advantage:

  Crack identification methods relying on visual inspection leverage the advantage of being adaptable to various lighting conditions. When adequately illuminated, these methods provide clear visibility, making it easier to detect surface cracks.

- Surface and Internal Crack Detection:

  While some techniques focus on surface cracks, others, particularly advanced non-destructive testing methods like ultrasonic testing and radiographic testing, excel in detecting both surface and internal cracks, ensuring a comprehensive assessment of material integrity.

- Algorithmic Precision:

  Automated systems equipped with sophisticated algorithms offer high precision in crack detection. With continuous advancements in machine learning and computer vision, the risk of false positives and negatives is decreasing, enhancing overall reliability.

- Material-Specific Effectiveness:

  Certain crack detection methods are tailored to specific materials, resulting in heightened effectiveness. This specificity ensures accurate identification and assessment, making the system adaptable to diverse materials and applications.

- Enhanced Sensitivity:

  Advances in non-destructive testing technologies have led to increased sensitivity, allowing for the detection of smaller and early-stage cracks. This heightened sensitivity ensures that even subtle structural issues are identified early on.

- Adaptability to Complex Structures:

  The evolution of inspection technologies has enabled better adaptability to complex structures. Innovative methods, such as robotic inspections and advanced imaging techniques, enhance the capability to navigate intricate geometries effectively.

- Efficiency in Automated Processes:

  Automated crack identification systems significantly reduce the time required for inspections, particularly in large structures or expansive areas. This efficiency results in timely assessments, minimizing the risk of overlooking critical cracks.

- Investment in Equipment and Training Pays Off:

  While certain non-destructive testing methods may require significant initial investment

in equipment and training, the long-term benefits outweigh the costs. The accuracy and reliability of these systems contribute to improved safety and reduced maintenance expenses over time.

- Microcrack Detection Capabilities:

  Cutting-edge technologies, including high-resolution imaging and advanced sensors, enable the effective detection of microcracks. This capability is crucial for preventing the escalation of subtle issues into more significant structural concerns.

- Enhanced Objectivity through Automation:

  Automated crack identification systems reduce human subjectivity, providing more consistent and objective results. Algorithms follow predefined criteria, minimizing variability and enhancing the overall reliability of inspections.

CHAPTER 4

SYSTEM DESIGN

4.1 HIGH LEVEL DESIGN (ARCHITECTURAL)

❖ FLOWCHART

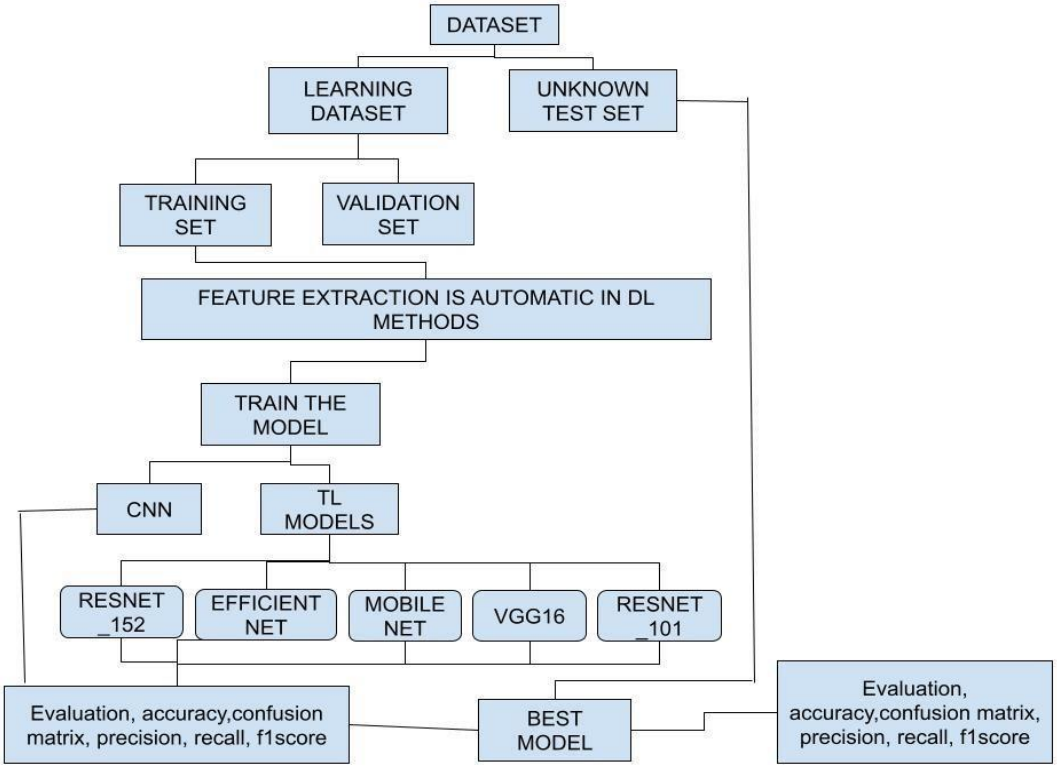

Fig 1: Flow Diagram for the models

We collected the images of 4 different types of cracks from various sources and saved it in different folders. The dataset was divided into learning dataset and test dataset. The training dataset is split into training set and validation set. Since we used deep learning technique the extraction of features is automatic. To train the model we used the below algorithms and selected the best model based on the accuracy and evaluation metrics of each algorithm. for the best models we gained the confusion matrix and the classification report.

❖ **CNN multi-class Architecture diagram**

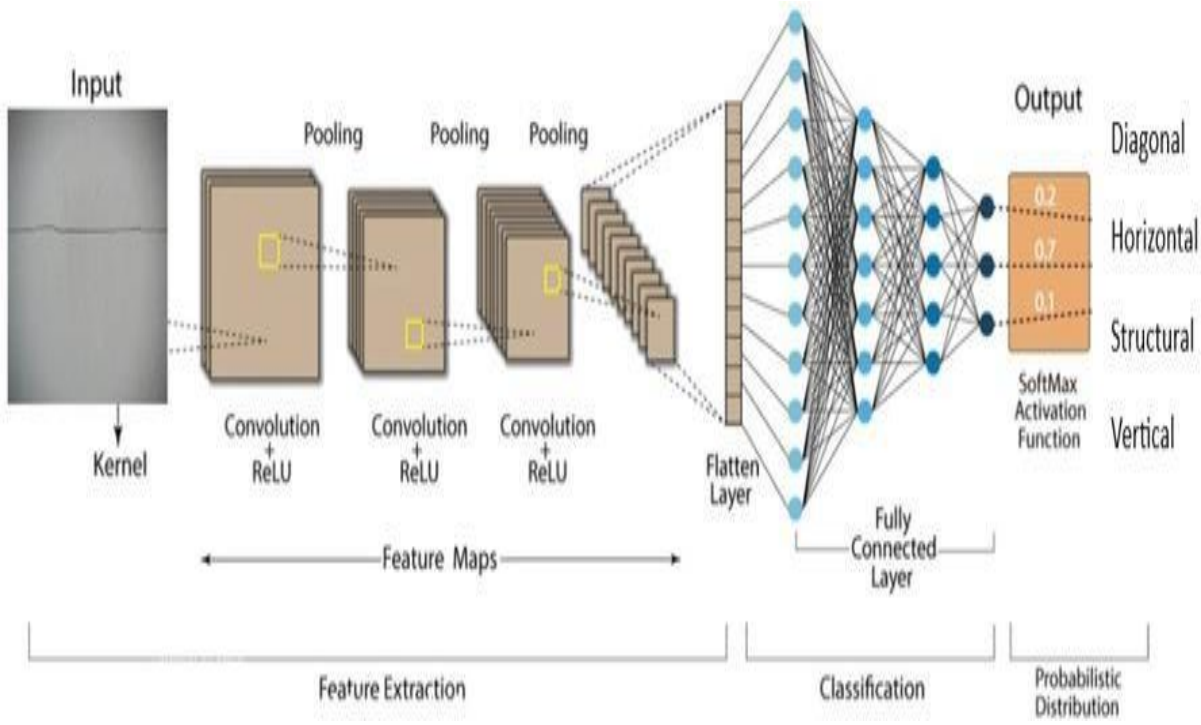

Fig 2: CNN Architecture for multi-class classification

A Convolutional Neural Network (CNN) architecture for multi-class classification typically consists of several layers designed to extract hierarchical features from input images and make predictions across multiple categories. At its core, the architecture comprises convolutional layers, which apply filters to input images to detect features such as edges, textures, and patterns. These convolutional layers are often followed by activation functions like ReLU (Rectified Linear Unit) to introduce non-linearity and pooling layers to reduce spatial dimensions while preserving important features. Subsequent convolutional and pooling layers progressively extract more abstract features. The final layers typically include fully connected layers, which take the output from the convolutional layers and map them to class labels using techniques like softmax activation for multi-class classification. Dropout layers may be added to prevent overfitting, and batch normalization layers can help stabilize and accelerate training. Overall, the CNN architecture for multi-class classification integrates these components to learn complex representations of input images and effectively distinguish between different classes.

❖ **Transfer Learning Architecture diagram**

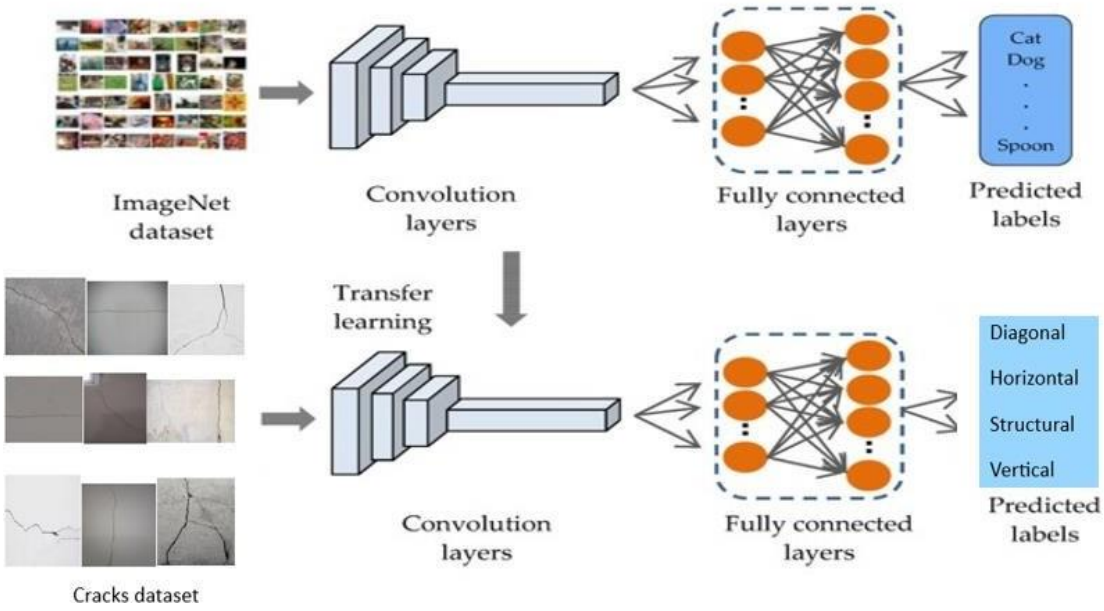

Fig 3: Transfer Learning Model Architecture for multi class classification

Transfer learning architecture leverages pre-trained models, such as VGG, ResNet, or Inception, which have been trained on large datasets like ImageNet. The process involves fine-tuning these models on a smaller data set specific to the target task. The architecture typically begins with the convolutional base of the pre-trained model, which consists of multiple convolutional layers responsible for feature extraction. These layers are frozen to retain the learned features. Next, additional convolutional layers may be added to adapt the model to the new dataset, followed by pooling layers to reduce dimensionality. Fully connected layers are then appended to the architecture to map the extracted features to the desired output classes. Finally, the output layer employs a SoftMax activation function to produce predicted labels for classification tasks. This architecture facilitates efficient training on smaller datasets by leveraging the knowledge gained from the pre-trained model while adapting to the specifics of the target task.

CHAPTER 5

DATA COLLECTION AND PREPARATION

5.1 DATA SOURCES

DATASET DETAILS: We collected images of cracks from various sources. For train dataset images were collected from sources like google and from cracks dataset available on Kaggle whereas for test dataset real time images were collected for the purpose of cracks classification using deep learning techniques.

- No. of Training Images = 411
- No. of Testing Images = 123

The training dataset includes images from the dataset named surface crack detection which was downloaded from Kaggle website and few are downloaded from google.
For test dataset we collected real time images of cracks from Reva university building.

- There are 4 types of cracks identified and segregated – vertical cracks, horizontal cracks, diagonal cracks, and structural cracks.
- In both training dataset and test dataset the collected images are classified into all the 4 types of cracks and segregated into different folders.

SAMPLE TRAINING DATASET:

Diagonal

Fig 4: Sample train image 1

Horizontal

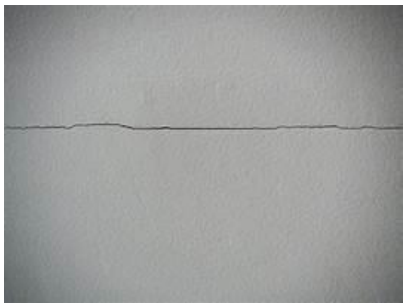

Fig 5: Sample train image 2

Structural

Fig 6: Sample train image 3

Vertical

Fig 7: Sample train image 4

SAMPLE TEST DATASET:

Diagonal

Fig 8: Sample test image 1

Horizontal

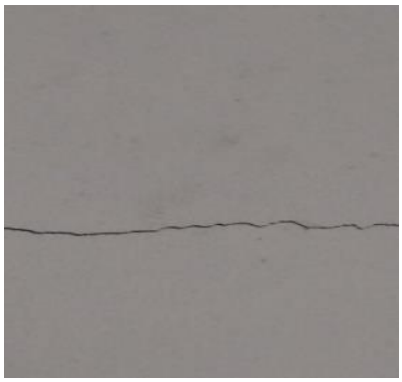

Fig 9: Sample test image 2

Structural

Fig 10: Sample test image 3

Vertical

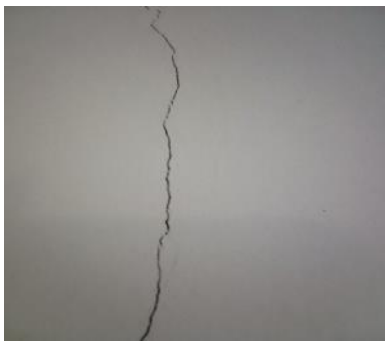

Fig 11: Sample test image 4

5.2 DATA PROFILING

Data profiling is a process within data management that involves the examination and analysis of datasets to gain a comprehensive understanding of their structure, content, and quality. The primary goal of data profiling is to uncover patterns, anomalies, and inconsistencies within the data, allowing organizations to make informed decisions about data quality, integration, and transformation. During data profiling, various statistical and descriptive techniques are applied to assess factors such as data distribution, completeness, accuracy, and uniqueness. This process helps identify data quality issues early in the data lifecycle, enabling organizations to implement effective data cleansing, transformation, or enrichment strategies.

In essence, data profiling serves as a foundational step in the data preparation phase, providing data analysts, scientists, and engineers with valuable insights into the characteristics of the datasets they are working with. By understanding the nuances of the data, organizations can enhance data governance, ensure compliance with regulatory requirements, and ultimately improve the reliability and usefulness of their data assets. For the analysis, we used only quality images and resized all the images such that all the images from the training and the test data are same.

CHAPTER 6

EXPLORATORY DATA ANALYSIS

6.1 DATA VISUALIZATION TECHNIQUES

Some available data visualization techniques are:

i. Bar Charts: Represent data with rectangular bars where the length corresponds to the value.

ii. Line Charts: Display data points connected by lines, often used to show trends over time.

iii. Pie Charts: Illustrate the proportion of parts to a whole using slices of a circle.

iv. Scatter Plots: Show individual data points on a two-dimensional graph to reveal relationships between variables.

v. Heatmaps: Visualize data in a matrix using colors to represent values, useful for large datasets.

vi. Histograms: Display the distribution of a single variable by dividing data into bins.

vii. Bubble Charts: Extend scatter plots by adding a third dimension represented by the size of markers.

viii. Box Plots: Summarize the distribution of a dataset, showing quartiles and outliers.

ix. Tables: The categorical values and the numerical values will be available in the form of rows and columns.

x. Waterfall Charts: Illustrate how an initial value is influenced by various positive and negative contributing factors.

For the project we mainly used 3 types of data visualization techniques – Tables, Line chart and Bar chart.

- Tables: We used tables to analyze the values in the form of rows and columns. Tables are used for classification report, accuracy report, etc.
- Line chart: To find the trend / growth of the accuracy, loss.
- Bar chart: To compare the growth between the train and test accuracy.

6.2 UNIVARIATE AND BIVARIATE ANALYSIS

Table 4: Model and test accuracy value table

Model	Accuracy (%)
CNN	98.37
RESNET_152_MODEL	75.56
EfficientnetB7 Model	73.33
MobilenetV3	73.33
VGG16	72.22
RESNET_101_MODEL	71.11

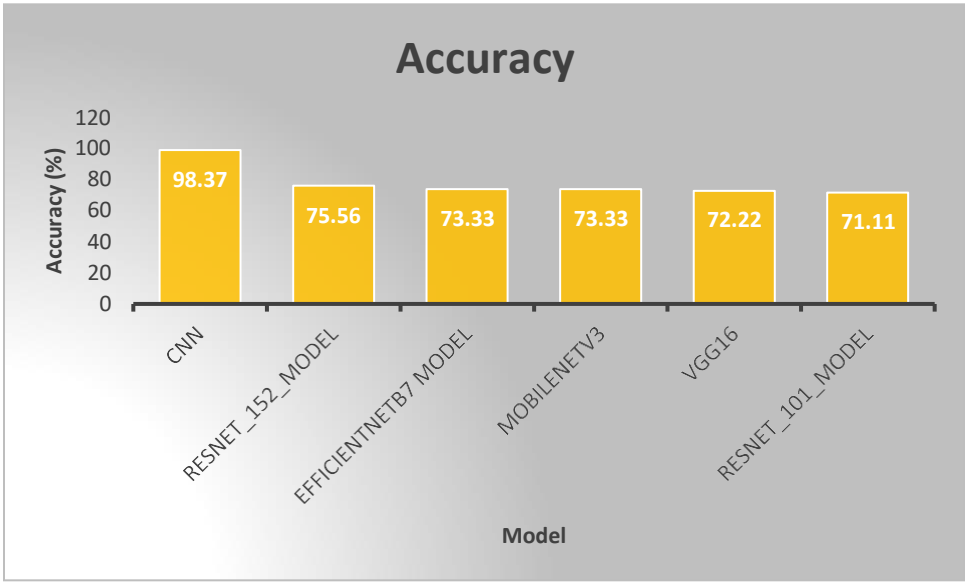

Fig 12: Univariate Analysis of model and accuracy

The above bar chart is a type of data visualisation technique. It shows the test accuracy for 6 models- CNN, Resnet_152_model, EfficientnetB7 model, MobilenetV3, VGG16, Resnet_101 model. The highest accuracy is given by CNN with 98.37% rather than Transfer learning models. In transfer learning models the highest accuracy is given by Resnet_152 model – 75.56% and the least accuracy is given by Resnet_101 model - 71.11%.

CHAPTER 7

METHODOLOGY

7.1 DATA MODELS

We collected the images of 4 different types of cracks from various sources and saved it in different folders. The dataset was divided into learning dataset and test dataset. The training dataset is split into training set and validation set. Since we used deep learning technique the extraction of features is automatic. To train the model we used the below algorithms and selected the best model based on the accuracy and evaluation metrics of each algorithm. for the best models we gained the confusion matrix and the classification report.

DATASET DETAILS: We collected images of cracks from various sources. For train dataset images were collected from sources like google and from cracks dataset available on Kaggle whereas for test dataset real time images were collected for the purpose of cracks classification using deep learning techniques.

- No. of Training Images = 411
- No. of Testing Images = 123

The training dataset includes images from the dataset named surface crack detection which was downloaded from Kaggle website and few are downloaded from google.
For test dataset we collected real time images of cracks from Reva university building.

- There are 4 types of cracks identified and segregated – vertical cracks , horizontal cracks , diagonal cracks and structural cracks.
- In both training dataset and test dataset the collected images are classified into all the 4 types of cracks and segregated into different folders.

Data profiling serves as a foundational step in the data preparation phase, providing data analysts, scientists, and engineers with valuable insights into the characteristics of the datasets they are working with. By understanding the nuances of the data, organizations can enhance data governance, ensure compliance with regulatory requirements, and ultimately improve the reliability and usefulness of their data assets. For the analysis, we used only quality images and resized all the images such that all the images from the training and the test data are same.

7.2 MODEL SELECTION

To train the model we used the below algorithms and selected the best model based on the accuracy and evaluation metrics of each algorithm. For the best models we gained the confusion matrix and the classification report.

- CNN model
- DenseNet_121 Model
- EfficientnetB7 Model
- iv) MobileNet_V3 Model
- VGG_16 Model
- ResNet_152 Model

CNN model:

This is Convolutional Neural Network models are widely used for image recognition tasks. CNNs consist of convolutional layers that learn to extract features from images hierarchically. CNN architectures can vary significantly in terms of depth, width, and connectivity patterns, allowing for flexibility in designing models for specific tasks.

ResNet_152 Model:

ResNet (Residual Network) is a deep CNN architecture that introduced the concept of residual learning, which helps alleviate the vanishing gradient problem in very deep networks. ResNet_152 is a variant with 152 layers, capable of achieving high accuracy on various image recognition tasks.

- ResNet_152 is a highly deep variant with 152 layers, offering increased representational capacity and enabling state-of-the-art performance on image classification tasks. It has a skip connection architecture where the output of one layer is added to the output of a deeper layer, facilitating the flow of gradients during training.

EfficientNetB7 Model:

EfficientNet is a family of CNN architectures designed for optimal performance in terms of accuracy and computational efficiency. EfficientNetB7 is one of the largest variants in the EfficientNet series, offering high accuracy while being relatively computationally efficient.

- They use a compound scaling method that balances network depth, width, and resolution to optimize both accuracy and efficiency. EfficientNetB7 is the largest variant in the

EfficientNet series, providing state-of-the-art performance on various image recognition benchmarks.

MobileNet_V3 Model:

MobileNet is a family of lightweight CNN architectures optimized for deployment on mobile and embedded devices. MobileNet_V3 is the third iteration of the MobileNet architecture, incorporating improvements in terms of both accuracy and efficiency compared to its predecessors.

- MobileNet_V3 incorporates inverted residuals and linear bottlenecks to improve accuracy while maintaining efficiency. It introduces multiple architecture improvements over previous versions, including Squeeze-and-Excitation blocks and hard-swish activation functions.

VGG_16 Model:

VGG (Visual Geometry Group) is a classic CNN architecture known for its simplicity and effectiveness. VGG_16 specifically refers to a VGG model with 16 layers, consisting mainly of 3x3 convolutional layers with max-pooling layers.

- VGG_16 has 16 weight layers, making it deeper than earlier models like AlexNet but shallower compared to more recent architectures. Despite its simplicity, VGG_16 achieves competitive performance on various image recognition tasks and serves as a strong baseline architecture.

RESNET_101_MODEL:

- ResNet_101 specifically refers to a ResNet model with 101 layers. Unlike traditional architectures, ResNet utilizes residual blocks, allowing the model to learn residual functions and mitigate the vanishing gradient problem.

- With 101 weight layers, ResNet_101 is exceptionally deep, surpassing the depth of earlier models. This depth enhances the model's ability to capture intricate features and representations, leading to superior performance on diverse computer vision tasks. ResNet_101 is known for its efficiency in training and impressive accuracy. The incorporation of residual blocks empowers the model to tackle challenges associated with training very deep networks, making it a popular and effective choice for various image recognition applications.

7.3 MODEL BUILDING

1. Convolutional Layers:

The model begins with a sequence of convolutional layers, each of which applies the Rectified Linear Unit (ReLU) activation function and collects features using 3x3 filters. Each convolutional layer is followed by two max-pooling layers (2x2), which lower the data's spatial dimensionality.

2. Flatten Layer:

This step prepares the multidimensional output for the fully connected layers by converting it into a one-dimensional array after the convolutional layers.

3. Fully Connected Layers:

The output that has been flattened is then routed through layers that are dense and completely linked. 512 neurons with ReLU activation make up the first dense layer. To reduce overfitting, a dropout layer with a dropout rate of 0.5 comes next. For additional feature extraction, a dropout layer, and an additional dense layer with 256 neurons are implemented.

4. Output Layer:

Using a softmax activation function, the final dense layer has three neurons, one for each of the three classes. This results in probability ratings for each class.

5. Compilation:

The RMSprop optimizer for brain tumor and Adam for Breast cancer are used to build the model, and accuracy is used as the evaluation metric. A learning rate of 0.001 is also used.

6. Training:

Using training data produced by a generator (train_generator), the model is trained for 50 epochs. To reduce the categorical crossentropy loss on the training data, the model's parameters are optimized throughout the training phase.

7. Validation:

To ensure that the model is not overfitting and to keep an eye on its capacity for generalization, its performance is assessed at each epoch on a different validation dataset (X_test, y_test).

- **CNN Model:**

This code defines a Convolutional Neural Network (CNN) using TensorFlow's Keras API for image classification. It consists of several convolutional and max-pooling layers, followed by flattening and dense layers. The model is compiled with the Adam optimizer, sparse categorical crossentropy loss, and accuracy metric. It is then trained on a dataset (train_ds) with specified batch size, validation data (val_ds), and runs for 100 epochs. The model architecture includes resizing, rescaling, and six convolutional layers with increasing filter sizes, ending with two dense layers.

- **ResNet_152 Model:**

This code defines and trains a neural network using the ResNet_152 architecture for image classification. It uses the pre-trained ResNet_152 as a base model, adds additional layers (Flatten, Dense) for feature extraction, and then trains the model on a given dataset using data augmentation and one-hot encoding for labels.

-Key parameters include image size, number of Dense layers, optimizer ('adam'), loss function ('categorical_crossentropy'), and training epochs (10).

- **EfficientNet_B7 Model:**

This code defines and trains a neural network using the EfficientNet_B7 architecture for image classification. It uses the pre-trained EfficientNet_B7as a base model, adds additional layers (Flatten, Dense) for feature extraction, and then trains the model on a given dataset using data augmentation and one-hot encoding for labels.

-Key parameters include image size, number of Dense layers, optimizer ('adam'), loss function ('categorical_crossentropy'), and training epochs (10).

- **MobileNet_V3 Model:**

This code defines and trains a neural network using the MobileNet_V3 architecture for image classification. It uses the pre-trained MobileNet_V3 as a base model, adds additional layers (Flatten, Dense) for feature extraction, and then trains the model on a given dataset using data augmentation and one-hot encoding for labels.

-Key parameters include image size, number of Dense layers, optimizer ('adam'), loss function ('categorical_crossentropy'), and training epochs (10).

- **VGG16 Model:**

This code defines and trains a neural network using the VGG16 architecture for image classification. It uses the pre-trained VGG16 as a base model, adds additional layers (Flatten, Dense) for feature extraction, and then trains the model on a given dataset using data augmentation and one-hot encoding for labels.

-Key parameters include image size, number of Dense layers, optimizer ('adam'), loss function ('categorical_crossentropy'), and training epochs (10).

❖ **ResNet_101_model:**

This code defines and trains a neural network using the ResNet_101 architecture for image classification. It uses the pre-trained ResNet_101 as a base model, adds additional layers (Flatten, Dense) for feature extraction, and then trains the model on a given dataset using data augmentation and one-hot encoding for labels.

-Key parameters include image size, number of Dense layers, optimizer ('adam'), loss function ('categorical_crossentropy'), and training epochs (10).

7.4 RESULTS

➢ **CNN:**

Accuracy:

Train accuracy: 99.55%; Validation accuracy: 98.67%

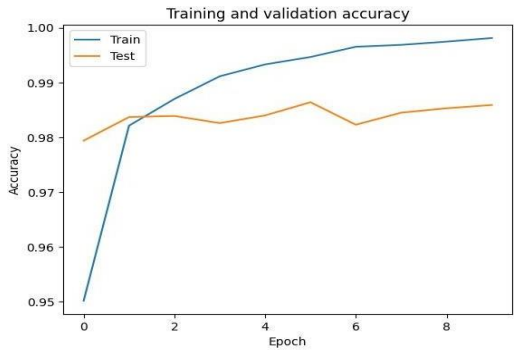

Fig13: CNN training and validation accuracy

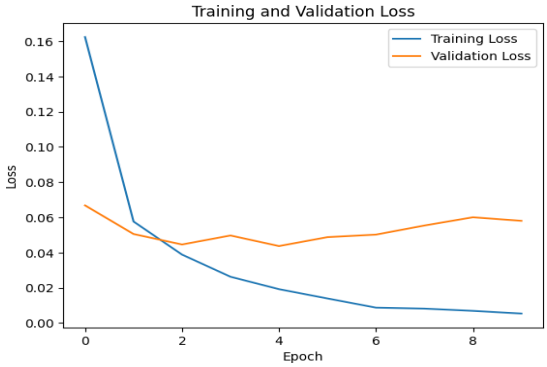

Fig14: CNN training and validation loss

➢ **TRANSFER LEARNING MODELS:**

✓ **Resnet_152_model:**

Accuracy:

Train accuracy: 77.81% ; Test accuracy: 75.56%

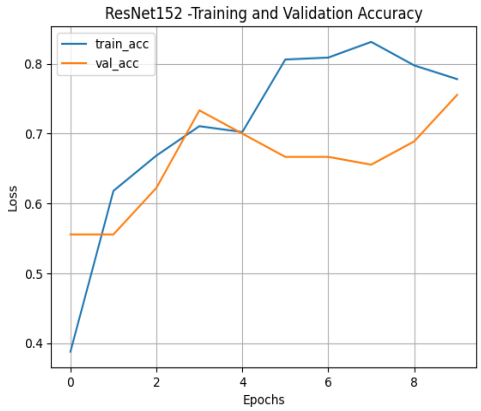

Fig15: Training and validation accuracy

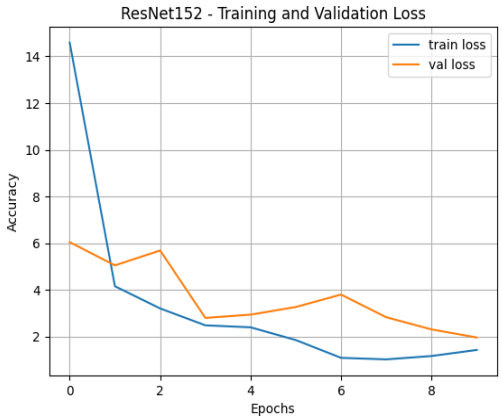

Fig16: Training and validation loss

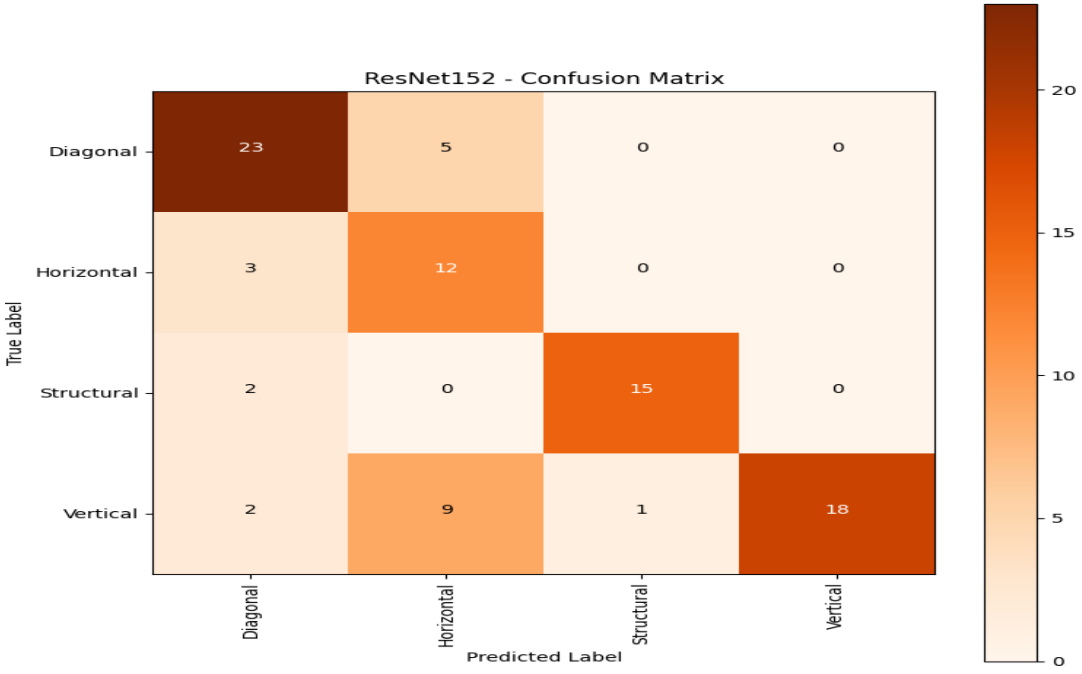

Fig17: Confusion matrix for Resnet_152

Table 5: Classification report for Resnet_152

	Precision	recall	F1-score
Diagonal	0.77	0.82	0.79
Horizontal	0.46	0.80	0.59
Structural	0.94	0.88	0.91
Vertical	1.00	0.60	0.75
Accuracy			0.76
Macro avg	0.79	0.78	0.76
Weighted avg	0.83	0.76	0.77

✓ **EfficientnetB7model:**

Accuracy:

Train accuracy: 83.43% ; Validation accuracy: 73.33%

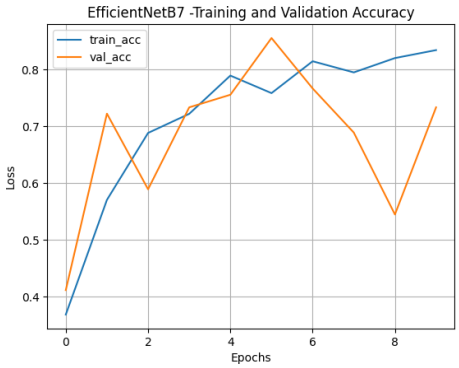

Fig 18: Training and validation accuracy

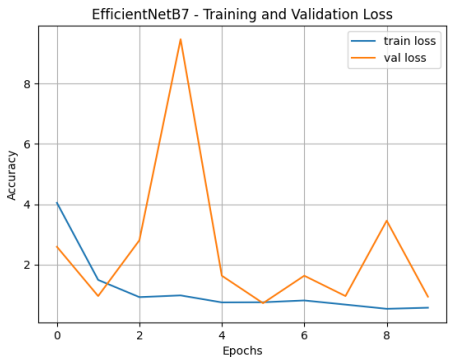

Fig 19: Training and validation loss

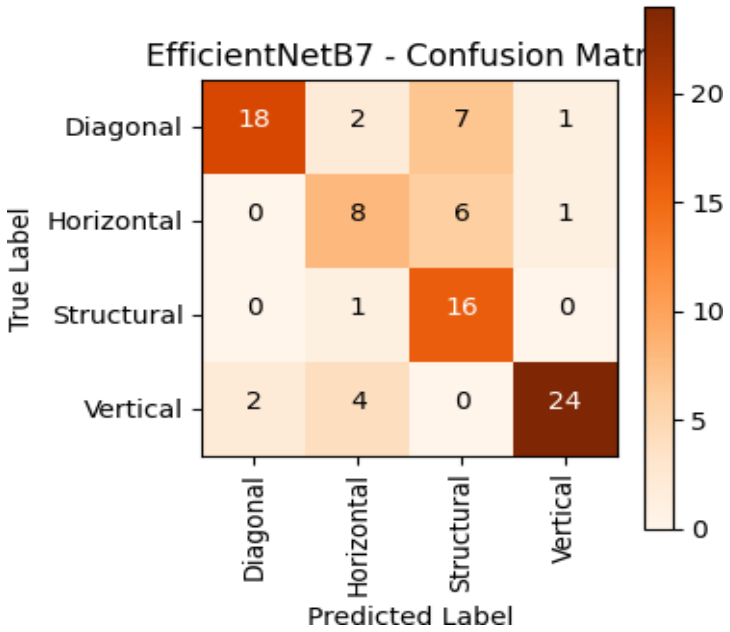

Fig 20: Confusion matrix for EfficientNetB7

Table 6: Classification report for EfficientNetB7

	Precision	**recall**	**F1-score**
Diagonal	0.90	0.64	0.75
Horizontal	0.53	0.53	0.53
Structural	0.55	0.94	0.70
Vertical	0.92	0.80	0.86
Accuracy			0.73
Macro avg	0.73	0.73	0.71
Weighted avg	0.78	0.73	0.74

✓ **MobilenetV3:**

Accuracy:

Train accuracy: 86.24% ; Validation accuracy: 73.33%

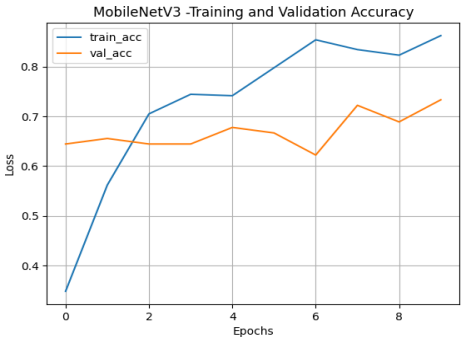

Fig 21: Training and validation accuracy

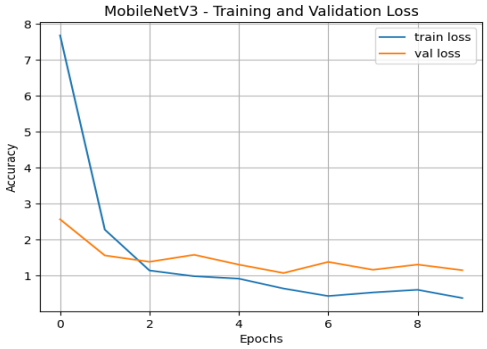

Fig 22: Training and validation loss

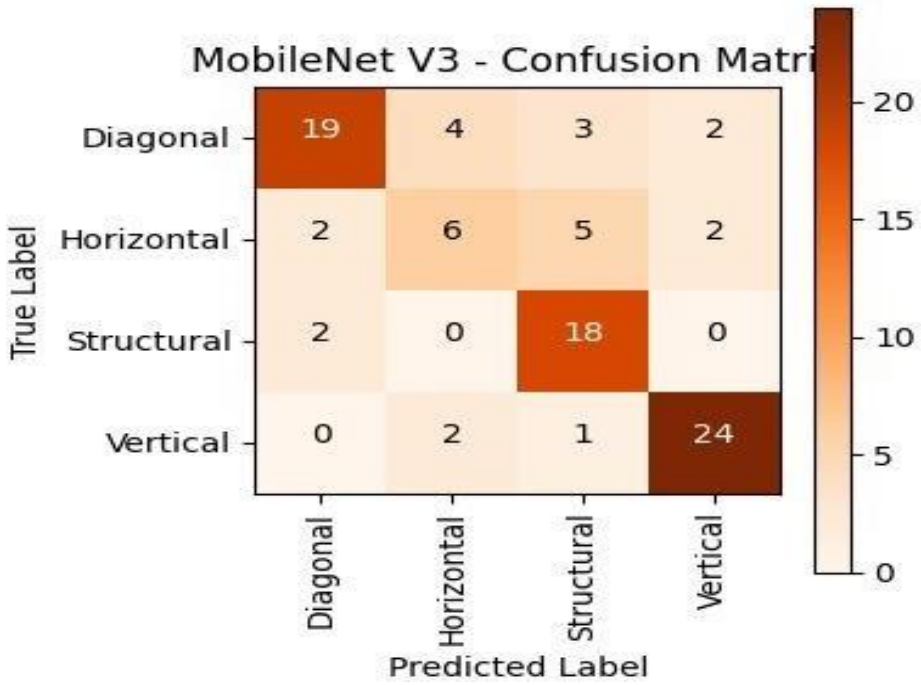

Fig 23: Confusion matrix of MobileNetV3

Table 7: Classification report for MobileNetV3

	Precision	recall	F1-score
Diagonal	0.94	0.57	0.71
Horizontal	0.46	0.87	0.60
Structural	0.85	1.00	0.92
Vertical	0.80	0.67	0.73
Accuracy			0.73
Macro avg	0.76	0.78	0.74
Weighted avg	0.80	0.73	0.74

✓ **VGG16:**

Accuracy:

Train accuracy: 83.71% ; Validation accuracy: 72.22%

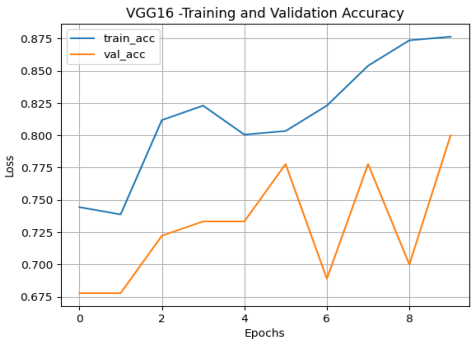

Fig 24: Training and validation accuracy

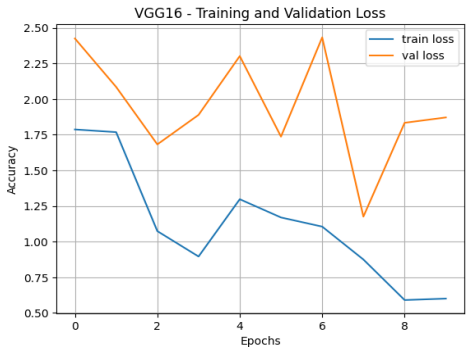

Fig 25: Training and validation loss

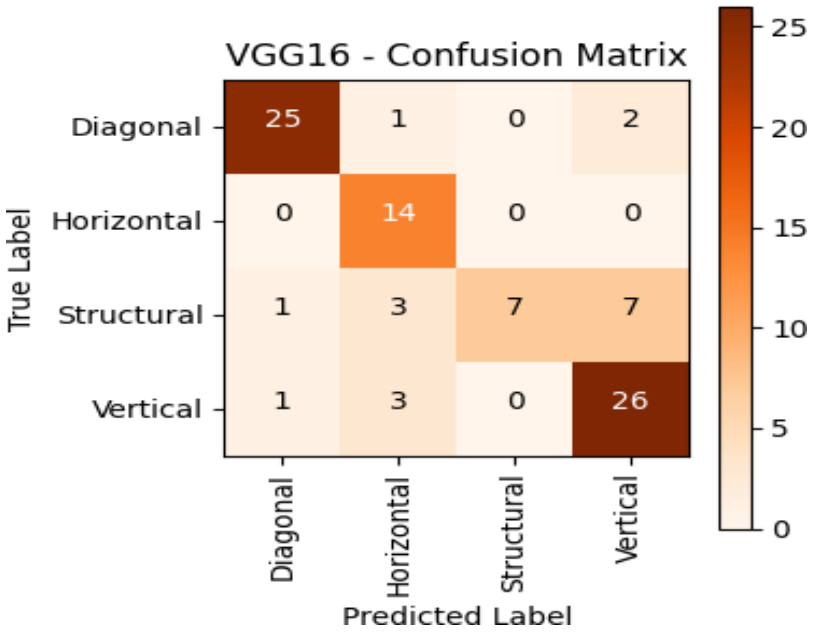

Fig 26: Confusion matrix for VGG16

Table 8: Classification report for VGG16

	Precision	**recall**	**F1-score**
Diagonal	0.93	0.89	0.91
Horizontal	0.67	1.00	0.80
Structural	1.00	0.39	0.56
Vertical	0.74	0.87	0.80
Accuracy			0.80
Macro avg	0.83	0.79	0.77
Weighted avg	0.84	0.80	0.79

✓ **Resnet_101_model:**

Accuracy:

Train accuracy: 83.71% ; Validation accuracy: 71.11%

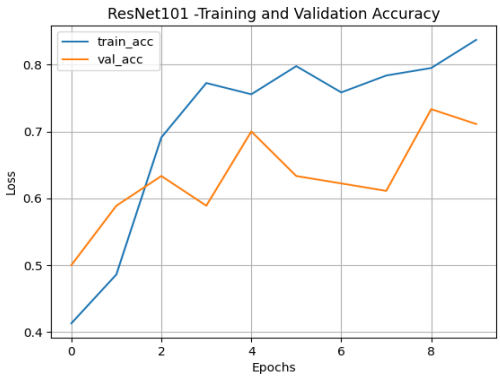

Fig 27: Training and validation accuracy

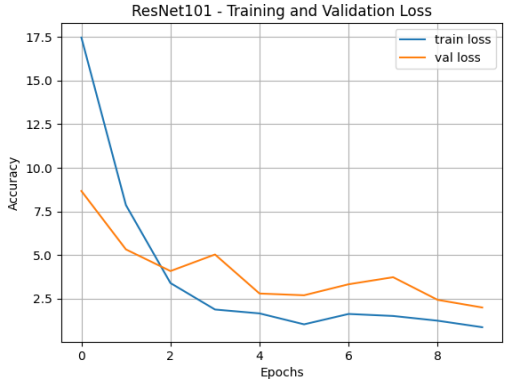

Fig 28: Training and validation loss

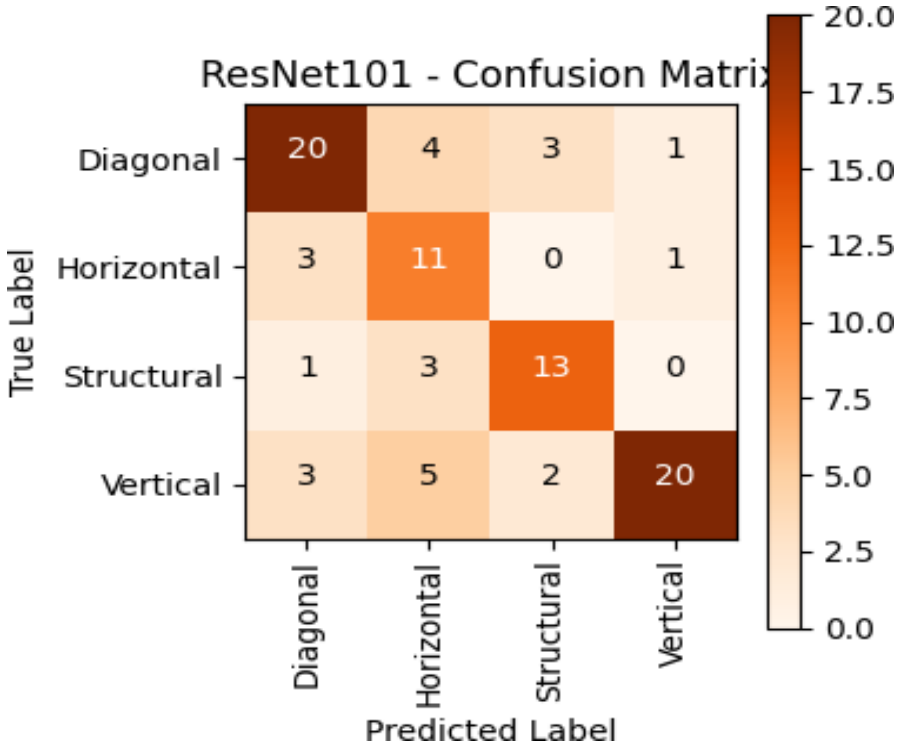

Fig 29: Confusion matrix for ResNet_101

Table 9: Classification report for Resnet_101

	Precision	**recall**	**F1-score**
Diagonal	0.74	0.71	0.73
Horizontal	0.48	0.73	0.58
Structural	0.72	0.76	0.74
Vertical	0.91	0.67	0.77
Accuracy			0.71
Macro avg	0.71	0.72	0.70
Weighted avg	0.75	0.71	0.72

Table 10: Transfer model with accuracy table:

SL. No	Transfer Learning MODEL	Train Accuracy (%)	Test Accuracy (%)
1.	CNN	99.82	98.41
1.	RESNET_152_MODEL	77.81	75.56
2.	EFFICIENTNETB7 MODEL	83.43	73.33
3.	MOBILENETV3	86.24	72.22
4.	VGG16	83.71	72.22
5.	RESNET_101_MODEL	83.71	71.11

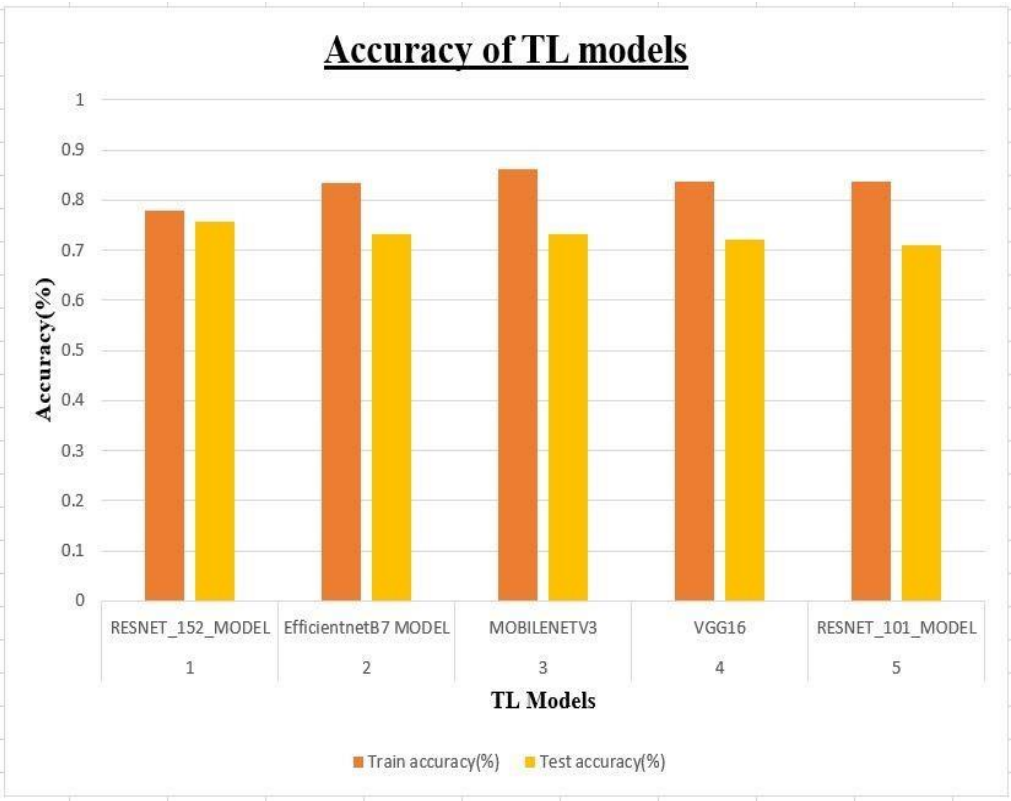

Fig 30: Bar chart for Transfer Learning models with accuracy

Comparison between the models:

- **CNN model:**

The CNN model represents a foundational architecture in the field of computer vision, offering versatility and applicability across a wide range of tasks. Its hierarchical structure allows it to automatically learn and extract features from raw pixel data, making it highly effective for tasks like image classification, object detection, and segmentation. However, the design of CNN architectures can vary significantly, with differences in depth, width, and connectivity patterns, leading to variations in computational efficiency, memory requirements, and performance. While CNNs provide a strong foundation for many computer vision tasks, they may require substantial computational resources, especially for deeper architectures with many parameters.

- **ResNet_152 Model:**

ResNet_152 revolutionized deep learning with its introduction of residual connections, enabling the training of extremely deep neural networks without encountering vanishing or exploding gradients. This architecture facilitates the capture of highly abstract features from images, leading to state-of-the-art performance on image recognition tasks. The skip

connections in ResNet_152 promote gradient flow during training, accelerating convergence and improving generalization, particularly in very deep networks. While ResNet_152 excels in terms of performance and scalability, training and deploying such deep models may require substantial computational resources and infrastructure.

- **EfficientNetB7 Model:**

EfficientNetB7 represents the epitome of efficiency-performance trade-offs in deep learning models. By systematically scaling network depth, width, and resolution, EfficientNet achieves state-of-the-art performance while maintaining computational efficiency. This balance is crucial for deployment in resource-constrained environments, such as mobile and embedded devices, where both accuracy and efficiency are paramount. The compound scaling method employed by EfficientNet ensures that the model achieves optimal performance across different scales, making it a versatile choice for a wide range of image recognition tasks. However, training EfficientNetB7 models may require significant computational resources due to their larger size and complexity.

- **MobileNet_V3 Model:**

MobileNet_V3 exemplifies the pursuit of lightweight architectures tailored for mobile and embedded deployment. Building upon the success of previous iterations, MobileNet_V3 incorporates innovative architectural designs, such as inverted residuals and linear bottlenecks, to improve both accuracy and computational efficiency. These optimizations make MobileNet_V3 well-suited for real-time applications on devices with limited resources, such as smartphones and IoT devices. However, while MobileNet_V3 achieves commendable performance in constrained environments, it may not match the accuracy of larger, more computationally intensive models like DenseNet or ResNet.

- **VGG_16 Model:**

VGG_16, with its simplicity and effectiveness, serves as a benchmark for understanding and evaluating deep learning architectures. Its straightforward design, consisting primarily of stacked 3x3 convolutional layers with max-pooling, makes it easy to implement and modify. While VGG_16 achieves competitive performance on various image recognition tasks, it may suffer from computational inefficiency due to its depth and large number of parameters. Nonetheless, its simplicity and transparency make it an attractive choice for educational purposes and as a baseline model for comparison with more complex architectures.

- **ResNet_101 Model:**

ResNet-101, like its predecessor ResNet-152, revolutionized deep learning by introducing residual connections that alleviate vanishing/exploding gradients. With skip connections enhancing gradient flow, ResNet-101 enables efficient training of deep networks, capturing intricate image features and achieving top-tier performance in image recognition tasks. While excelling in scalability and performance, deploying ResNet-101 may demand significant computational resources and infrastructure for training and deployment.

❖ **Generative Adversarial Network (GAN):**

GANs are used for generating realistic synthetic data, such as images or videos, with applications in image synthesis, style transfer, and data augmentation. Using GAN images are generated for 4 types of cracks which are as follows:

Diagonal:

Fig 31: Real images from the dataset

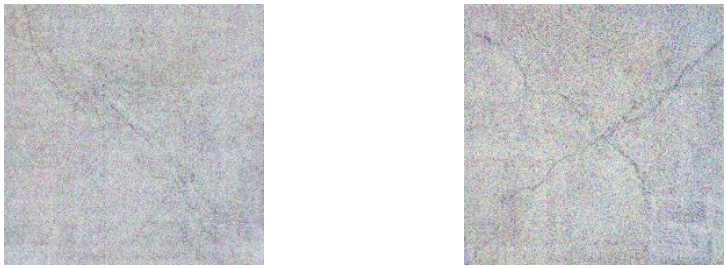

Fig 32: Generated images by GAN

Vertical:

Fig 33: Real images from the dataset

Fig 34: Generated images by GAN

Structural:

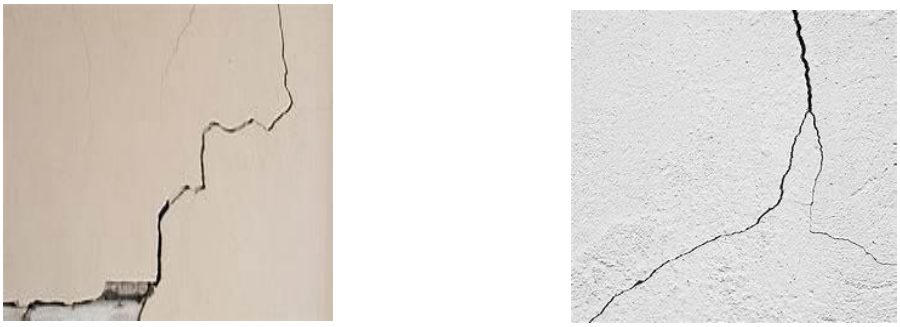

Fig 35: Real images from the dataset

Fig 36: Generated images by GAN

Horizontal:

Fig 37: Real images from the dataset

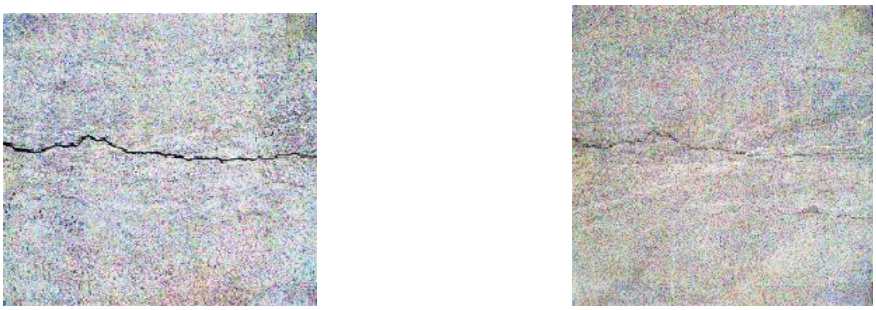

Fig 38: Generated images by GAN

CHAPTER 8

TESTING

- **CNN model**

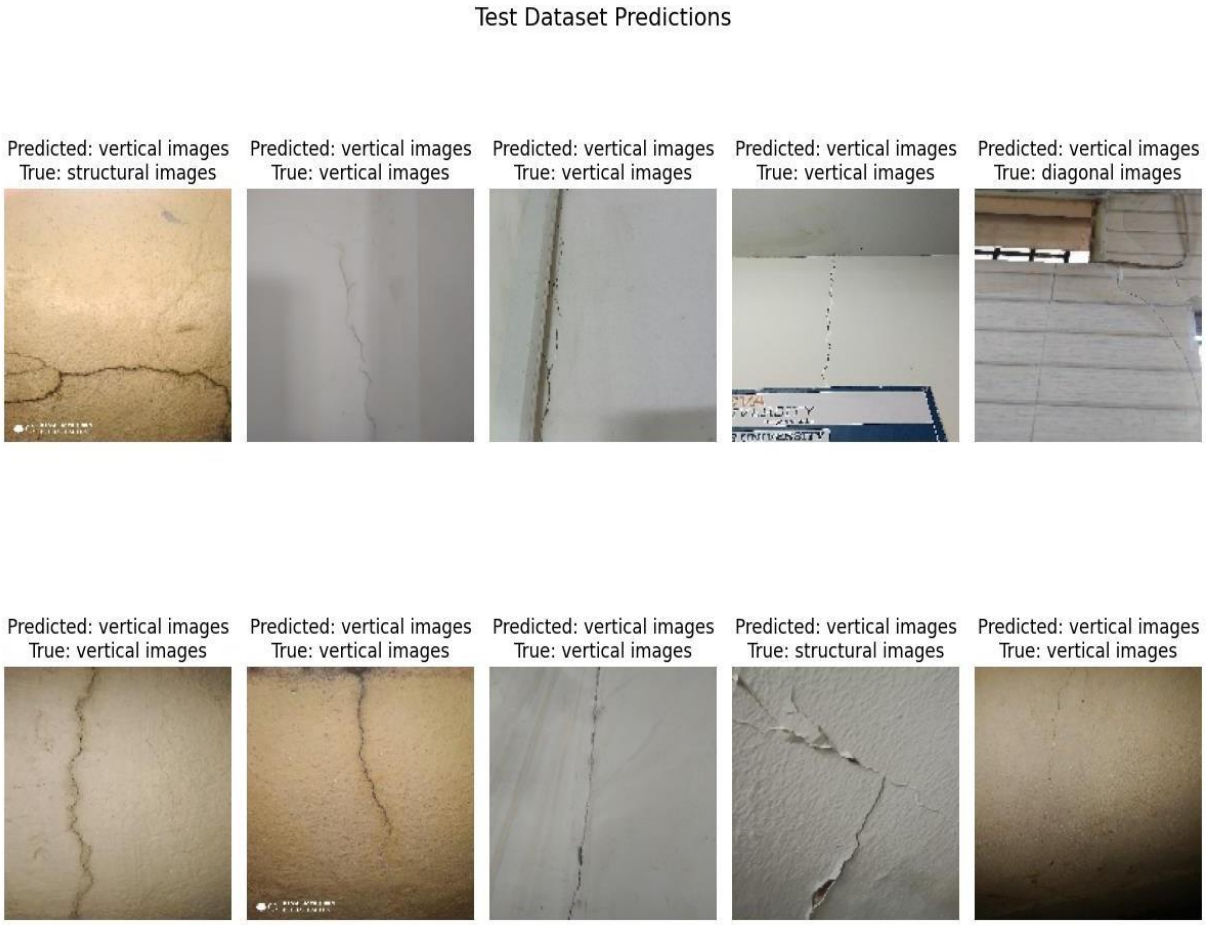

Fig 39: Test dataset predictions by CNN

The above are the predictions done by CNN model for the test dataset. Out of 10 images for different types the true predictions are 7 and the false predictions are 3. The accuracy percentage in this case is 70%.

- **Resnet_152_model**

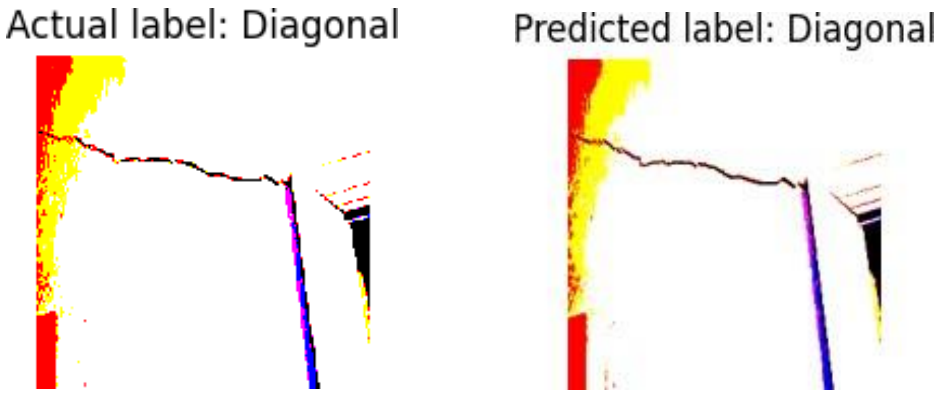

Fig 40: True predictions for Resnet_152

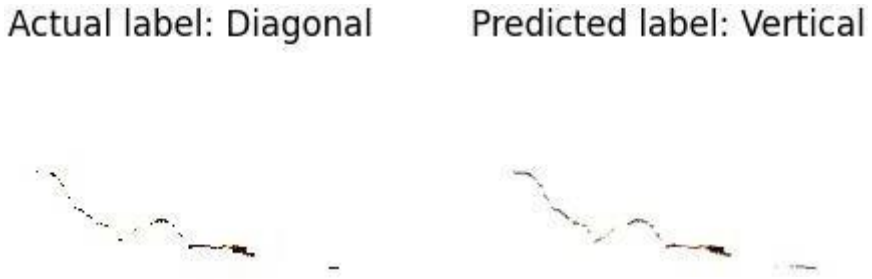

Fig 41: False predictions for Resnet_152

- **EfficientnetB7model**

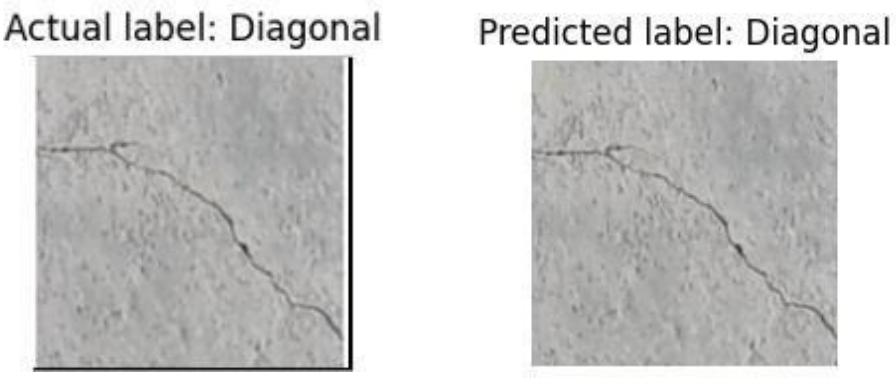

Fig 42: True predictions for EfficientnetB7

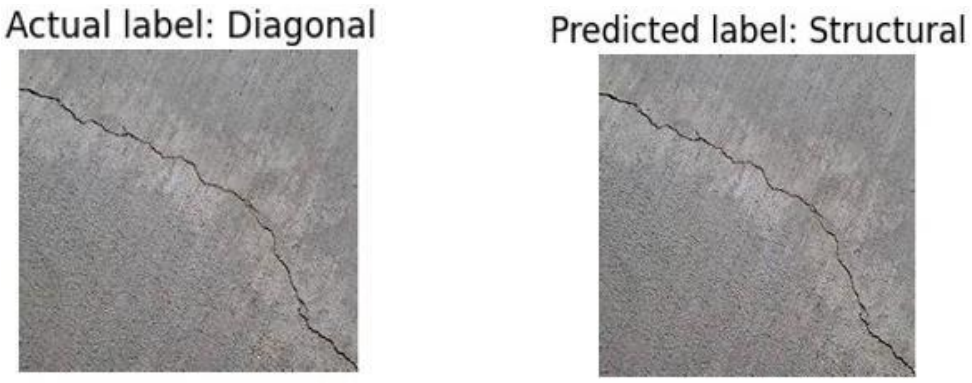

Fig 43: False predictions for EfficientnetB7

- **MobilenetV3**

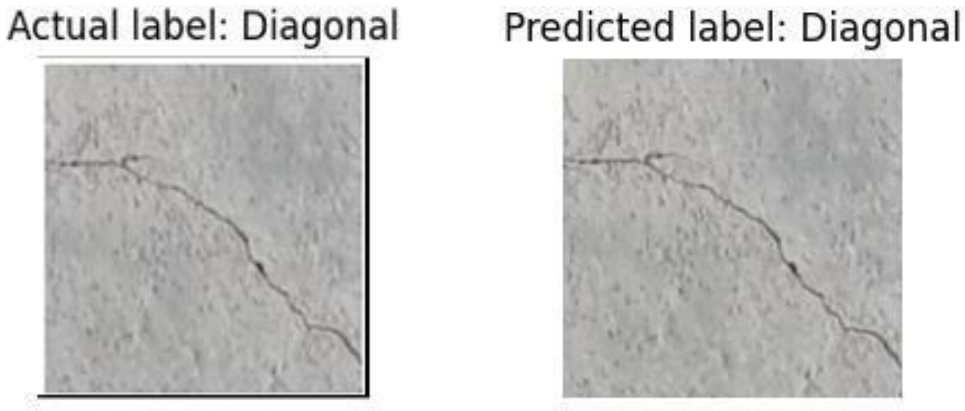

Fig 44: True predictions for MobilenetV3

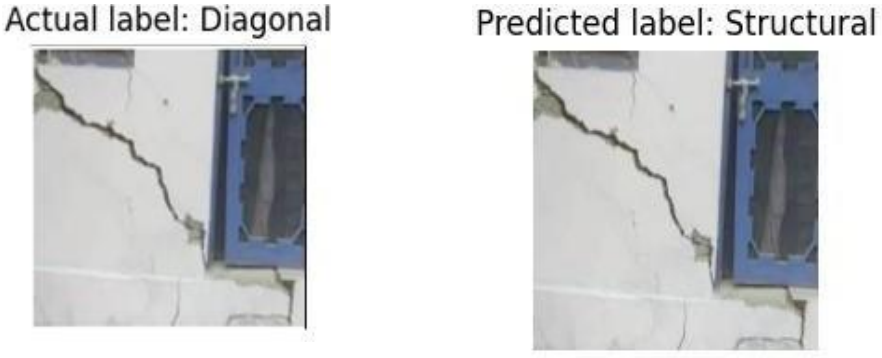

Fig 45: False predictions for MobilenetV3

- **Resnet_101_model**

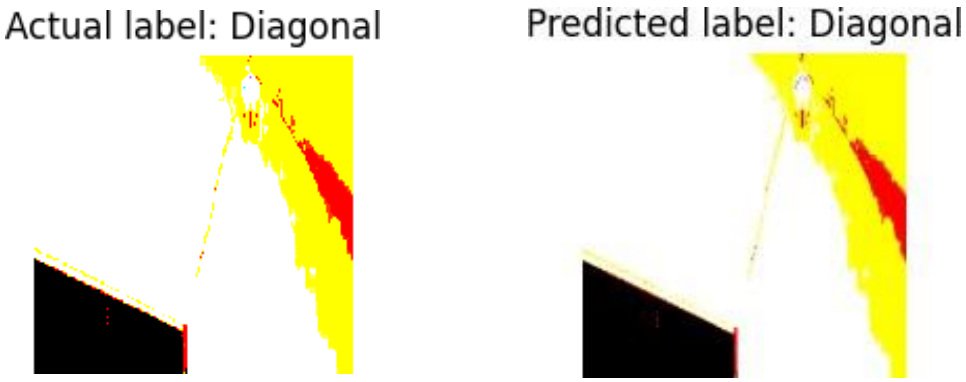

Fig 46: True prediction for Resnet_101

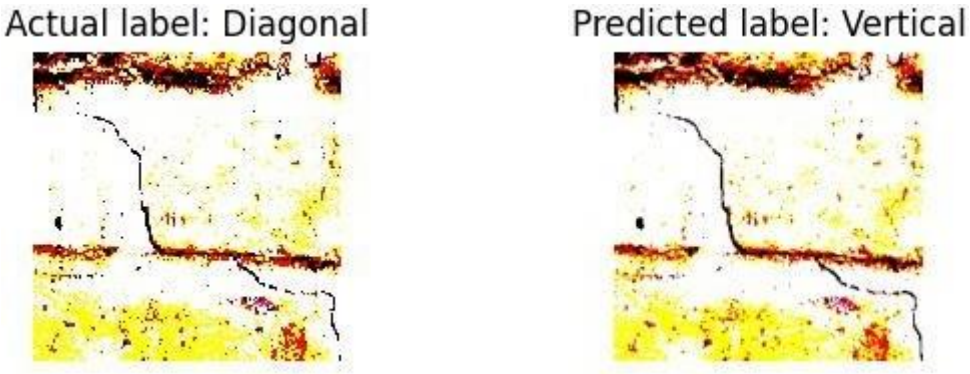

Fig 47: False predictions for Resnet_101

SCOPE OF FUTURE ENHANCEMENT:

We will develop another binary model to compare the generated images with the real images.

CHAPTER 9

CONCLUSION

The integration of a deep learning-based crack detection system stands as a transformative force in bolstering infrastructure safety through the automated identification of structural cracks. This pioneering approach, centered on the utilization of Convolutional Neural Networks (CNNs), has demonstrated remarkable accuracy, marking a significant stride towards proactive interventions in the maintenance and preservation of critical assets.

By automating crack detection processes, CNNs enable the swift and precise identification of structural issues. This not only expedites the maintenance workflow but also ensures that potential problems are addressed promptly, contributing to the overall safety and longevity of infrastructure. The utilization of CNNs showcases the capacity of deep learning models to discern intricate patterns associated with crack formations, surpassing traditional methods. The project sets a noteworthy precedent for the adoption of such cutting-edge technologies in safeguarding the integrity of our built environment.

However, the resource constraints, specifically limitations in processing power that hindered the acquisition of desired images from Generative Adversarial Networks (GANs), posed challenges to the project's full potential. Despite these limitations, the accomplishments achieved with the available resources are significant, emphasizing the resilience and adaptability of the implemented solution.

The acknowledgment of the potential to generate better images with enhanced resources implies a trajectory of continuous improvement. As technology evolves and resources become more accessible, there lies a promising avenue for further refinement and expansion of the deep learning-based crack detection system. The ongoing commitment to innovation, coupled with an anticipation of overcoming current limitations, positions the project as a foundational stepping stone towards a more resilient and technologically advanced approach to infrastructure safety.

CHAPTER 10

BIBLIOGRAPHY

[1] Danajitha K K. etal., "Detection of Cracks in High Rise Buildings using Drones", Proceedings of the Third International Conference on Electronics and Sustainable Communication Systems (ICESC 2022) IEEE Xplore DOI:10.1109/ICESC54411.2022.9885251

[2] Rafiul Bari Angan .etal., "Health Monitoring of Old Buildings in Bangladesh: Detection of Cracks and Dampness Using Image Processing," 2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)16-17 June 2023 DOI: 10.1109/NCIM59001.2023.10212708

[3] Dr. M . Ravichand.etal., "Crack on Brick Wall Detection by Computer Vision using Machine Learning," Proceedings of the Sixth International Conference on Electronics, Communication and Aerospace Technology (ICECA 2022) IEEE Xplore DOI: 10.1109/ICECA55336.2022.10009343

[4] Angela Busheska.etal., "Machine Learning and Thermography Applied to the Detection and Classification of Cracks in Buildings," 2023 IEEE Conference on Technologies for Sustainability (SusTech) DOI: 10.1109/SusTech57309.2023.10129614

[5] Prashanth Kannadaguli.etal., "Microwave Imaging based Automatic Crack Detection System using Machine Learning for Columns," IEEE international conference, DOI: 10:10.1109/CSNT.2020.02

[6] R. Thandaiah Prabu .etal., "Automated Crack and Damage Identification in Premises using Aerial Images based on Machine Learning Techniques," Proceedings of the Sixth International Conference on I-SMAC, IEEE Xplore, DOI: 10.1109/I-SMAC55078.2022.9987391

[7] Theres Davies.etal., "A Novel Method for Concrete Crack Detection Using Image Processing Technique," DOI: 10.1109/ACCTHPA57160.2023.10083343

[8] Nhat-Duc Hoang.etal., "Image Processing-Based Recognition of Wall Defects Using Machine Learning Approaches and Steerable Filters," Hindawi Computational Intelligence and Neuroscience Volume 2018, Article ID 7913952, https://doi.org/10.1155/2018/7913952

[9] P. Padmapoorani.etal., "Application of machine learning for crack detection on concrete structures using CNN architecture," ISSN 1517-7076 articles e20230010, 2023

[10] Min Jae Park.etal., “Machine Learning-Based Concrete Crack Depth Prediction Using Thermal Images Taken under Daylight Conditions,” Remote Sens. 2022, 14, 2151, doi.org/10.3390/rs14092151
[11] Md. Monirul Islam.etal., “CNN Based on Transfer Learning Models Using Data Augmentation and Transformation for Detection of Concrete Crack,” Algorithms 2022, 15, 287, doi.org/10.3390/a15080287
[12] Fahim Ullah.etal., “Inspecting Buildings Using Drones and Computer Vision: A Machine Learning Approach to Detect Cracks and Damages,” Drones 2022, 6, 5. doi.org/10.3390/drones6010005
[13] Mayank Sharma.etal., “Concrete Crack Detection Using the Integration of Convolutional Neural Network and Support Vector Machine,” Article · June 2018 DOI: 10.14456/scitechasia.2018.11
[14] Enrique Valero.etal., “Automated defect detection and classification in ashlar masonry walls using machine learning,” doi.org/10.1016/j.autcon.2019.102846
[15] Priyanka Gupta.etal., “Performance Evaluation of Deep Learning Models For Surface Crack Detection,” 1st IEEE International conference on Innovations in High-Speed Communication and Signal Processing (IEEE-IHCSP) 4-5 March, 2023, DOI: 10.1109/IHCSP56702.2023.10127175
[16] Feng Wang.etal., “Application of Deep Learning Algorithm in Crack Detection of Green Building Materials,” 2022 6th Asian Conference on Artificial Intelligence Technology (ACAIT), DOI: 10.1109/ACAIT56212.2022.10137849
[17] B. Swaminathan.etal., “A Deep Learning-based Approach for Surface Crack Detection using Convolutional Neural Network,” Proceedings of the International Conference on Edge Computing and Applications (ICECAA 2022), IEEE Xplore, DOI: 10.1109/ICECAA55415.2022.9936270
[18] Bubryur Kim.etal., “Surface crack detection using deep learning with shallow CNN architecture for enhanced computation,” Neural Computing and Applications, doi.org/10.1007/s00521-021-05690-8
[19] Shuwei Li.etal., “Detection of concealed cracks from ground penetrating radar images based on deep learning algorithm,” Construction and Building Materials 273 (2021), doi.org/10.1016/j.conbuildmat.2020.121949
[20] Kaiwen Chen.etal., “Automated crack segmentation in close-range building façade inspection images using deep learning techniques,” Journal of Building Engineering 43 (2021), doi.org/10.1016/j.jobe.2021.102913

Online Resources:

URL:
https://www.kaggle.com/code/mahendra77/vegetable-image-classification-using-cnn
https://www.geeksforgeeks.org/multiclass-image-classification-using-transfer-learning/
https://www.analyticsvidhya.com/blog/2021/04/generate-your-own-dataset-using-gan/

YOUTUBE LINKS:
https://youtu.be/Q1NC3NbmVlc?si=NeRWHtVTrhjWS64p

CHAPTER 11

APPENDIX – Sample Source Code/Pseudo Code

Code for CNN model:

```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import cv2

import os

os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"

import warnings

warnings.filterwarnings('ignore')

from sklearn.metrics import confusion_matrix, classification_report

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Activation, BatchNormalization, Conv2D, Dense,
Dropout, Flatten, MaxPooling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.losses import CategoricalCrossentropy

from tensorflow.keras.regularizers import l2

from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

def create_model():

    model = Sequential([
```

```
        Conv2D(filters=128, kernel_size=(5, 5), padding='valid',
    input_shape=(IMG_WIDTH, IMG_HEIGHT, 3)),

        Activation('relu'),

        MaxPooling2D(pool_size=(2, 2)),

        BatchNormalization(),


        Conv2D(filters=64, kernel_size=(3, 3), padding='valid',
    kernel_regularizer=l2(0.00005)),

        Activation('relu'),

        MaxPooling2D(pool_size=(2, 2)),

        BatchNormalization(),


        Conv2D(filters=32, kernel_size=(3, 3), padding='valid',
    kernel_regularizer=l2(0.00005)),

        Activation('relu'),

        MaxPooling2D(pool_size=(2, 2)),

        BatchNormalization(),


        Flatten(),


        Dense(units=256, activation='relu'),

        Dropout(0.5),

        Dense(units=6, activation='softmax')

    ])

    return model
```

Transfer learning model code (Resnet_152 Model):

```
import numpy as np

import os

import cv2

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix, classification_report

from keras.applications.resnet import ResNet152, preprocess_input

from keras.layers import Dense, Flatten

from keras.models import Model

from keras.preprocessing.image import ImageDataGenerator

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2,
random_state=42)

# Perform data augmentation on the training set

train_datagen = ImageDataGenerator(rotation_range=20, width_shift_range=0.1,
height_shift_range=0.1,

                                   shear_range=0.1, zoom_range=0.2, horizontal_flip=True,
fill_mode='nearest')

train_datagen.fit(X_train)

# Convert labels to one-hot encoding

y_train_onehot = np.eye(num_classes, dtype=np.int32)[y_train]

y_test_onehot = np.eye(num_classes, dtype=np.int32)[y_test]

# Train the model

history = model.fit(train_datagen.flow(X_train, y_train_onehot, batch_size=32),
epochs=10,

                    validation_data=(X_test, y_test_onehot))
```

Transfer learning code (EfficientnetB7 model):

```
import numpy as np

import os

import cv2

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix, classification_report

from keras.applications.efficientnet import EfficientNetB7, preprocess_input

from keras.layers import Dense, Flatten

from keras.models import Model

from keras.preprocessing.image import ImageDataGenerator

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2,
random_state=42)

# Perform data augmentation on the training set

train_datagen = ImageDataGenerator(rotation_range=20, width_shift_range=0.1,
height_shift_range=0.1,

                                   shear_range=0.1, zoom_range=0.2, horizontal_flip=True,
fill_mode='nearest')

train_datagen.fit(X_train)

# Convert labels to one-hot encoding

y_train_onehot = np.eye(num_classes, dtype=np.int32)[y_train]

y_test_onehot = np.eye(num_classes, dtype=np.int32)[y_test]

# Train the model

history = model.fit(train_datagen.flow(X_train, y_train_onehot, batch_size=16),
epochs=10,

                    validation_data=(X_test, y_test_onehot))
```