# *BONE FRACTURE CLASSIFICATION THROUGH EDGE DETECTION USING DL IN MEDICAL IMAGING*

*A Project report submitted in the partial fulfillment of the requirements for the award of the degree of*

**Bachelor of Technology**
In
**Computer Science & Engineering**

Submitted by


| | |
|---|---|
| **G.POOJITHA** | **(21KD1A0550)** |
| **G.SHIVA** | **(21KD1A0554)** |
| **G.LAHARI** | **(21KD1A0555)** |
| **B.JAYA SAI CHANDER** | **(21KD1A0520)** |


Under the guidance of

**Dr.V. PRASAD, Ph.D.**

**Professor**

**Department of CSE**



**Department of Computer Science & Engineering**

**LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY**

**An Autonomous Institution**

*Affiliated to JNTU Gurajada, Vizianagaram, Approved by A.I.C.T.E,Accredited by NAAC with "A" Grade & NBA Jonnada, Vizianagaram Dist.535005.*


***2021-2025***

**LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY (A)**

*(Approved by A.I.C.T.E & Affiliated to JNTU Gurajada, Vizianagaram,*
*Accredited by NBA & Accredited by NAAC with 'A' GRADE)*
*JONNADA, DENKADA (M), VIZIANAGARAM - 535005.*

# BONAFIDE CERTIFICATE

This is to certify that the project entitled **"BONE FRACTURE CLASSIFICATION THROUGH EDGE DETECTION USING DL IN MEDICAL IMAGING"** is a Bonafide record of the work done by **G.POOJITHA (21KD1A0550), G.SHIVA (21KD1A0554), G.CHANDRA LAHARI (21KD1A0555), B.JAYASAI CHANDER (21KD1A0A520)** under the guidance of **Dr.V. PRASAD, Ph.D, Professor**. in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from Lendi Institute of Engineering and Technology(Affiliated to JNTU Gurajada), Jonnada, Vizianagaram for the year 2025.

**Internal Guide**

**Dr.V. PRASAD, Ph.D.**
**Professor**
**Department of C.S. E**

**Head of the Department**

**Dr. A. Rama Rao, M. Tech, Ph.D**
**Professor**
**Department of CSE**

**External Examiner**

# ACKNOWLEDGEMENT

With great solemnity and sincerity, we offer our profuse thanks to our management, for providing all the resources to complete our project successfully. We express our deepest sense of gratitude and pay our sincere thanks to our guide **Dr.V. PRASAD, Professor, Ph.D**. Department of Computer Science & Engineering, who evinced keen interest in our efforts and provided his valuable guidance throughout our project work.

We thank our project coordinator **Dr.V.ANJI REDDY, Professor**, Department of Computer Science & Engineering, who has made his support available in a number of ways and helped us to complete our project work in correct manner.

We thank our **Dr.A.RAMA RAO** Head of the Department of Computer Science & Engineering who helped us to complete our project work in a truthful method.

We thank our gratitude to our principal **Dr.V.V.RAMA REDDY**, for his kind attention and valuable guidance to us throughout this course in carrying out the project.

We wish to express gratitude to our Management Members who supported us in providing good lab facility.

We also thankful to All Staff Members of Department of Computer Science & Engineering, for helping us to complete this project work by giving valuable suggestions.

All of the above we greatfully acknowledge and express our thanks to our parents who have been instrumental for the success of this project which play a vital role.

**G.POOJITHA (21KD1A0550)**
**G.SHIVA (21KD1A0554)**
**G.LAHARI (21KD1A0555)**
**B.JAYA SAI CHANDER (21KD1A0520)**

# DECLARATION

We hereby declare that the project work entitled "**BONE FRACTURE CLASSIFICATION THROUGH EDGE DETECTION USING DL IN MEDICAL IMAGING**" submitted to the Lendi Institute of Engineering and Technology is a record of an original work done by **G.POOJITHA (21KD1A0550), G.SHIVA (21KD1A0554), G.CHANDRA LAHARI (21KD1A0555), B.JAYA SAI CHANDER (21KD1A0520)** under the guidance of **Dr.V. PRASAD, Professor, Ph.D**, Computer Science & Engineering, Lendi Institute of Engineering & Technology (Affiliated to JNTU Gurajada). This project work is submitted in the partial fulfillment of the requirements for the award of the degree Bachelor of Technology in Computer Science & Engineering. This entire project is done with the best of our knowledge and is not submitted to any university for the award of degree/diploma.

**G.POOJITHA (21KD1A0550)**
**G.SHIVA (21KD1A0554)**
**G.LAHARI (21KD1A0555)**
**B.JAYA SAI CHANDER (21KD1A0520)**

**LENDI INSTITUTE OF ENGINEERING & TECHNOLOGY (A)**

*(Approved by A.I.C.T.E & Affiliated to JNTU Gurajada, Vizianagaram,*
*Accredited by NBA & Accredited by NAAC with 'A' GRADE)*
*JONNADA, DENKADA (M), VIZIANAGARAM - 535005.*

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## VISION

To be a frontier in computing technologies to produce globally competent computer science engineering graduates with moral values to build a vibrant society and nation.

## MISSION

- Providing a strong theoretical and practical background in computer science engineering with an emphasis on software development.
- Inculcating professional behaviour, strong ethical values, innovative research capabilities, and leadership abilities.
- Imparting the technical skills necessary for continued learning towards their professional growth and contribution to society and rural communities.

## PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO-1: Graduates will have strong knowledge and skills to comprehend latest tools and techniques of Computer Engineering so that they can analyze, design and create computing products and solutions for real life problems.

PEO-2: Graduates shall have multidisciplinary approach, professional attitude and ethics, communication and teamwork skills, and an ability to relate and solve social issues through computer engineering.

PEO-3: Graduates will engage in life-long learning and professional development to adapt to rapidly changing technology.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1: Ability to grasp advanced programming techniques to solve contemporary issues.

PSO-2: Have knowledge and expertise to analyze data and networks using latest tools and technologies.

PSO-3: Qualify in national and international competitive examinations for successful higher studies and employment.

## PROGRAM OUTCOMES (POS)

**PO-1** Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO-2** Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO-3** Design/development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO-4** Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO-5** Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitation.

**PO-6** The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO-7** Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO-8** Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO-9** Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO-10** Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO-11** Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO-12** Life-Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological

# ABSTRACT

Diagnosing bone fractures from X-ray images has traditionally been a manual and time-consuming process. However, deep learning (DL) has introduced new opportunities for automating medical image analysis. This research presents a novel hybrid approach that integrates deep convolutional neural networks (ConvNet) with an enhanced Canny edge detection technique to improve bone fracture classification. The Canny edge detection is employed to extract essential image features by outlining object boundaries, effectively highlighting fracture regions. These edge-enhanced images are then input into a deep ConvNet for learning and classification, enhancing the model's capacity to detect relevant features. This integration improves both diagnostic accuracy and computational performance. An enhanced version of the Canny algorithm is used to ensure precise edge detection, which contributes significantly to the robustness of the model. The hybrid model was evaluated against advanced models such as ResNet using X-ray medical imaging datasets. Results demonstrated that the proposed model achieved superior performance with 98% classification accuracy, 98% precision, 98.5% recall, and a 98% F1-score. These findings indicate that incorporating edge-based feature extraction significantly enhances the effectiveness of deep ConvNet models. Overall, this hybrid deep learning framework offers a reliable, efficient, and accurate solution for automated bone fracture diagnosis, setting a new standard for combining traditional image processing with advanced neural networks.



**Keywords**: Deep Learning, Classification Models,Canny Edge Detection, Automated Bone Fracture Detection through Edge Detection, Analysis, Split Values.


**Outcomes:**Our project titled **"BONE FRACTURE CLASSIFICATION THROUGH EDGE DETECTION USING DL IN MEDICAL IMAGING"** is mapped with the following outcomes:

| | | |
|---|---|---|
| **Program Outcomes** | : | PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12. |
| **Program Specific Outcomes** | : | PSO1, PSO2, PSO3. |

# LIST OF CONTENTS

# LIST OF FIGURES



# LIST OF TABLES

# LIST OF OUPUT SCREENSHOTS

# CHAPTER 1

# INTRODUCTION

# INTRODUCTION

Bone fractures are among the most common injuries in the field of orthopedics, affecting millions worldwide every year. Accurate and timely detection of fractures is crucial for effective treatment and recovery. Traditionally, radiologists and orthopedic specialists rely on manual inspection of X-ray and CT scan images to diagnose fractures, which can be time-consuming and prone to human error, especially in cases of subtle or complex fractures. To overcome these challenges, computer-aided diagnosis (CAD) systems have emerged as powerful tools to support medical professionals in fracture detection and classification.

Recent advances in **deep learning** have revolutionized image-based diagnosis, offering high accuracy in detecting patterns and abnormalities in medical images. Deep learning models, particularly convolutional neural networks (CNNs), have demonstrated exceptional capability in automatic feature extraction and classification tasks. However, the presence of noise, varying image quality, and complex bone structures in medical imaging often demand robust preprocessing techniques to enhance model performance.

One such preprocessing technique is the **Canny Edge Detection** algorithm, which is widely used for identifying sharp changes in intensity, effectively highlighting the edges and contours of bone structures in radiographic images. By applying Canny edge detection before feeding the images into deep learning models, the system can better focus on the fracture lines and structural discontinuities, leading to improved detection accuracy.

This study proposes a hybrid approach that combines **Canny edge detection** with **deep learning models** to automate the process of bone fracture detection and classification. The system aims to preprocess bone images using edge detection, extract meaningful features through deep networks, and classify the type of fracture accurately. Such an automated system can not only enhance diagnostic precision but also reduce workload on healthcare professionals and expedite patient care.

The detection and classification of bone fractures is a critical task in the field of radiology and orthopedic medicine. Accurate identification of fractures can significantly impact the effectiveness of treatment and recovery. Traditionally, this task has relied on manual interpretation of X-ray and CT scan images, which can be time-consuming and subject to human error. As a result, there is a growing demand for automated systems that can assist healthcare professionals in diagnosing bone fractures more efficiently and accurately.

Deep learning (DL) techniques, particularly **Convolutional Neural Networks (CNNs)**, have proven to be highly effective in automating image classification tasks. These models are capable of learning hierarchical features from raw image data without the need for explicit feature engineering.

By leveraging large datasets of labeled medical images, DL models can learn to differentiate between various types of fractures and healthy bone structures, offering a reliable and consistent diagnostic tool.Incorporating **Canny edge detection** as a preprocessing step in this process helps highlight important structural edges in medical images, such as fracture lines and bone boundaries.

Canny edge detection enhances the feature extraction process, providing clearer input for deep learning models. This preprocessing step improves the model's ability to detect subtle fractures, which may otherwise be overlooked.

The combination of Canny edge detection and deep learning allows for more accurate and efficient classification of bone fractures. By automating this process, radiologists can focus on more complex cases and reduce the workload associated with routine imaging interpretation. This also has the potential to speed up diagnosis, enabling quicker treatment decisions and improving patient outcomes.

**Deep Learning**

Deep learning, a subset of artificial intelligence, uses neural networks—especially convolutional neural networks (CNNs)—to automatically learn and extract important features from large datasets. In the context of bone fracture classification, DL models can analyze complex patterns in medical images that may be difficult for the human eye to detect, especially in subtle or overlapping fractures.

By training on thousands of annotated X-ray or CT images, DL models can differentiate between normal bone structures and various types of fractures. They can classify fractures based on their pattern, such as transverse, oblique, spiral, comminuted, and greenstick fractures. This automated classification not only improves diagnostic accuracy but also assists medical professionals in planning effective treatment strategies.

In this project, deep learning is combined with preprocessing techniques like Canny edge detection to further enhance the quality of the input images. Canny edge detection highlights the edges and discontinuities in bone structures, making fracture lines more prominent. This helps the deep learning model to focus on relevant features, leading to higher classification performance.

The application of DL in bone fracture detection offers several advantages: faster diagnosis, reduced workload for radiologists, consistent results, and potential for deployment in remote areas lacking specialist doctors. Automated systems powered by deep learning can serve as reliable second opinions, supporting healthcare professionals in critical decision-making.This project aims to design a robust system that leverages the power of deep learning and edge detection techniques to detect and classify bone fractures accurately.

The ultimate goal is to build a smart, efficient, and scalable solution that enhances orthopedic diagnostics and contributes to better patient care worldwide.

Deep learning models, especially **Convolutional Neural Networks (CNNs)**, are designed to handle image data effectively. They learn spatial hierarchies of features, meaning they detect low-level features like edges in early layers and high-level features like fracture shapes in deeper layers.

One key strength of deep learning is its scalability. As more medical images are collected and annotated, the models continue to improve in accuracy and reliability. This makes deep learning an ideal solution for modern healthcare systems dealing with large volumes of imaging data.

Deep learning models also support **data augmentation** techniques, where the training data is artificially expanded through rotations, flips, and brightness changes. This helps the model generalize better to new, unseen images.

Moreover, deep learning frameworks such as TensorFlow and PyTorch have made it easier for researchers and developers to build, train, and deploy complex models with minimal coding.

**Canny Edge Detection**

Canny Edge Detection is one of the most popular and effective edge detection algorithms in the field of image processing and computer vision. Developed by John F. Canny in 1986, this algorithm is designed to detect a wide range of edges in images with high accuracy and minimal error. It has become a standard method due to its optimal performance in terms of detection, localization, and response.

The main purpose of edge detection is to identify points in an image where the brightness changes sharply, indicating the presence of object boundaries, shapes, or structural discontinuities. In medical imaging, especially in bone X-rays, edge detection is crucial as it helps to highlight fractures and differentiate bone structures from surrounding tissues.

Canny edge detection works through a series of well-defined steps. The first step is **noise reduction** using a Gaussian filter. Since medical images often contain noise that can cause false edges, smoothing the image helps in reducing such errors and prepares the image for more accurate processing.

Next, the algorithm calculates the **gradient intensity** of each pixel in the image. This is done using gradient operators like Sobel filters. The gradient represents how much the pixel intensity changes and in which direction, which is essential for detecting edges.Following this, the **non-maximum suppression** step is applied.

This process thins out the edges by retaining only the local maxima of the gradient magnitude. As a result, only the sharpest edges are kept while others are suppressed, providing a clean and precise edge map.

One of the unique features of Canny edge detection is its **double thresholding** mechanism. This involves setting two threshold values: a high threshold and a low threshold. Pixels with gradient values above the high threshold are considered strong edges, while those between the low and high thresholds are considered weak edges.In the final stage, known as **edge tracking by hysteresis**, weak edges are included only if they are connected to strong edges.

This step helps eliminate isolated weak edges that are likely caused by noise while preserving the continuity of real edges.The advantages of Canny edge detection include its ability to detect true edges with low error rate, accurate localization of edges, and the production of thin, well-defined edge lines. It is particularly effective in handling images with varying lighting conditions and noise levels.

In the context of bone fracture detection, Canny edge detection plays a critical role in preprocessing the medical images. By highlighting fracture lines and bone boundaries, it simplifies the task for deep learning models, allowing them to focus on meaningful features and ignore irrelevant details.

Moreover, this technique reduces computational load by emphasizing the most significant structures in the image. It helps in improving the overall performance of automated diagnosis systems, leading to higher detection and classification accuracy.Canny edge detection is widely used not only in medical imaging but also in applications like object recognition, face detection, and industrial inspection, proving its versatility and robustness.

Deep Convolutional Neural Networks (Deep ConvNets) are a class of deep learning models specifically designed for analyzing and classifying image data. They have become a cornerstone in medical image

analysis due to their high accuracy, scalability, and ability to automatically learn important features from raw images.In bone fracture detection.

Deep ConvNets play a crucial role by accurately identifying subtle cracks, lines, and discontinuities in X-ray or CT scan images that may be difficult for human eyes to detect, especially in early or complex fracture cases.The strength of ConvNets comes from their layered structure. Each layer in the network performs a specific task in transforming the raw pixel data into meaningful patterns.

At the core of the model are **convolutional layers**. These layers apply small filters (kernels) that scan the input image and extract local features. In medical images, these features include edges, contours, and texture variations that signify fracture lines or bone deformities.

Following convolution, the **activation function**, typically the Rectified Linear Unit (ReLU), is applied. ReLU introduces non-linearity into the model, enabling it to capture complex patterns beyond simple linear relationships.

**Pooling layers** come next, usually applying a technique called Max Pooling. This step reduces the size of the feature maps, making the model computationally efficient while retaining the most significant information. Pooling also helps make the model invariant to small shifts and distortions, which is useful since medical images may vary slightly in angle or quality.

As the network deepens, successive convolutional and pooling layers capture increasingly abstract and complex features. Early layers might detect simple edges, while deeper layers can recognize fracture patterns, bone segments, and structural anomalies.After multiple feature extraction layers, the data is flattened and passed to **fully connected layers**. These layers act like traditional neural networks, combining the extracted features to make final classification decisions.

The final layer of the network is the **output layer**, which uses activation functions like Softmax (for multi-class classification) or Sigmoid (for binary classification) to predict the probability of each fracture type. For example, it can classify whether the bone is normal or fractured, and if fractured, whether it is transverse, oblique, spiral, or comminuted.

Training a Deep ConvNet involves feeding it a large dataset of labeled images. The model learns by comparing its predictions with the ground truth and updating its weights through **backpropagation** and **gradient descent** algorithms. This process minimizes the error, improving classification accuracy over time.

In this project, Deep ConvNets are combined with **Canny edge detection** as a preprocessing step. Canny edge detection highlights the significant edges and contours in the image, such as fracture lines and bone boundaries. This helps the ConvNet focus on the most relevant features, leading to better performance.

One of the key advantages of Deep ConvNets is their ability to automatically extract features, eliminating the need for manual feature engineering. This makes them highly scalable and adaptable to different types of medical images and fracture classifications.

Deep ConvNets also support **transfer learning**, where pre-trained models like, ResNet, or DenseNet, which are trained on massive image datasets, can be fine-tuned for specific tasks like bone fracture detection. This saves time, reduces computational resources, and often improves accuracy.

In clinical practice, ConvNet-based systems can serve as decision-support tools. They provide a second opinion to radiologists, flagging potential fractures that might be overlooked, especially in busy or resource-limited healthcare settings.

Such systems improve diagnostic speed and consistency, reduce radiologist workload, and ensure patients receive timely and accurate treatment. They can also be deployed in remote areas where specialist doctors are not available.

Deep ConvNets have been successfully used in various medical fields, including tumor detection, pneumonia screening, and now, bone fracture classification. Their robustness and flexibility make them suitable for both 2D (X-ray) image analysis.

Despite their power, Deep ConvNets require large, high-quality datasets for training, and their performance depends on proper preprocessing, model tuning, and validation techniques.

In this project, the goal is to build a deep learning pipeline where Canny edge detection enhances image features, and a Deep ConvNet accurately detects and classifies fractures into their respective types.This integrated approach aims to deliver an automated, efficient, and scalable solution that benefits both healthcare professionals and patients by improving diagnostic precision and speed.

## 1.1 PROJECT OVERVIEW

This project focuses on the detection and classification of bone fractures using deep learning techniques combined with Canny edge detection. Bone fractures, if not accurately diagnosed, can lead to severe complications. Traditional diagnosis through manual observation of X-ray or CT images is time-consuming and prone to human error. To address this, the project proposes an automated system that enhances diagnostic accuracy and efficiency.The system uses **Canny edge detection** to preprocess images by highlighting fracture lines and bone edges, making important features more distinct. These preprocessed images are then fed into a **Deep Convolutional Neural Network (ConvNet)**, which automatically extracts features and classifies the images into fracture types.The ConvNet model learns from labeled medical images, distinguishing between normal bones and different fracture categories such as transverse, oblique, and comminuted fractures. This combination of edge detection and deep learning ensures both fine detail recognition and robust classification performance.By automating fracture detection, the system supports radiologists and orthopedic doctors in making faster and more accurate decisions.

Bone fracture classification plays a crucial role in orthopaedic diagnosis and treatment planning. This project aims to develop an automated system for classifying bone fractures using deep learning techniques integrated with edge detection methods in medical imaging. Specifically, the project leverages **Canny Edge Detection** to enhance fracture boundaries in X-ray images before feeding them into a deep learning model for classification.

The process begins with the acquisition of medical X-ray images, followed by **preprocessing** steps including grayscale conversion, noise reduction, and application of **Canny Edge Detection** to highlight fracture lines and structural boundaries. These enhanced images are then input to a deep learning model—such as **ResNet-50** or a custom CNN—which is trained to classify:**Presence** of a fracture (fractured or normal),**Bone type** (e.g., hand, elbow, shoulder),**Fracture severity** (mild, moderate, severe).

The integration of edge detection focuses the model on relevant structural features, improving its ability to detect subtle fractures and reducing background noise.This approach offers potential improvements in diagnostic speed and accuracy, assisting radiologists in clinical settings and supporting remote or automated fracture diagnosis.

## 1.2 PROJECT DELIVERABLES

This project will deliver a complete deep learning-based system for automatic bone fracture detection and classification. The key deliverables include a **preprocessing module** using Canny edge detection to enhance image features and highlight fracture lines. A trained **Deep Convolutional Neural Network (ConvNet)** model will be provided for accurate classification of bone fractures into various types.Additionally, the project will deliver a **dataset of labeled medical images** used for training and validation. A **user interface (optional)** will be developed for easy input of medical images and display of classification results. The deliverables also include detailed **project documentation** covering methodology, model architecture, training results, and performance evaluation.

The project will deliver a complete deep learning-based system for automated bone fracture classification using Canny Edge Detection on medical X-ray images. Key deliverables include: (1) a preprocessed and annotated dataset of hand, elbow, and shoulder X-rays with applied edge detection, (2) a trained deep learning model capable of classifying fracture presence, bone type, and severity, (3) performance evaluation metrics such as accuracy, precision, recall, and confusion matrices, (4) a user-friendly interface or script to test new X-ray images, and (5) a detailed project report documenting methodology, experiments, and results. Optionally, visualizations comparing raw images, edge-detected images, and classification outcomes will also be provided to demonstrate the system's interpretability.

The primary deliverable of this project is a deep learning-based system that can classify bone fractures using enhanced X-ray images processed through Canny Edge Detection. This system is designed to support the detection and categorization of fractures in hand, elbow, and shoulder bones. One of the first outputs is a fully preprocessed dataset that includes raw X-ray images, corresponding edge-detected versions, and labeled classifications for training and evaluation. Alongside this, a robust image preprocessing pipeline will be delivered, incorporating steps such as grayscale conversion, noise reduction using Gaussian blur, and application of Canny Edge Detection to highlight fracture boundaries more effectively.

## 1.3 PROJECT SCOPE

This project aims to develop an automated system for classifying bone fractures in medical images (like X-rays) by combining traditional image processing techniques (Canny edge detection) with deep learning models. The system will enhance bone edges using the Canny edge detection algorithm and then classify the type of fracture (or detect the absence of fracture) using deep learning classifiers such as Convolutional Neural Networks (CNNs).

The primary deliverable of this project is a deep learning-based system that can classify bone fractures using enhanced X-ray images processed through Canny Edge Detection. This system is designed to support the detection and categorization of fractures in hand, elbow, and shoulder bones. One of the first outputs is a fully preprocessed dataset that includes raw X-ray images, corresponding edge-detected versions, and labeled classifications for training and evaluation. Alongside this, a robust image preprocessing pipeline will be delivered, incorporating steps such as grayscale conversion, noise reduction using Gaussian blur, and application of Canny Edge Detection to highlight fracture boundaries more effectively.

The core of the system is a trained deep learning model—such as ResNet-50—fine-tuned for medical image classification. This model will be capable of determining whether a bone is fractured or normal, identifying the bone region (hand, elbow, or shoulder), and assessing the severity of the fracture (mild, moderate, or severe). The system will be evaluated using standard performance metrics including accuracy, precision, recall, F1-score, and confusion matrices. These results will be benchmarked to validate the impact of edge detection on model accuracy.

# CHAPTER 2

# LITERATURE SURVEY

# LITERATURE SURVEY

1. **Mohamed et al., "Explainable Transfer Learning-Based Deep Learning Model for Pelvis Fracture Detection"**

   This study introduces a transfer learning-based deep learning model for pelvis fracture detection, focusing on explainability. It aims to improve the interpretability of deep learning predictions in medical applications, specifically for pelvis fractures. The model helps radiologists understand the reasoning behind fracture detection, making it a valuable tool for clinical decision-making.

2. **Title: Hands-On Machine Learning with Scikit-Learn and TensorFlow**

   This book serves as a practical guide to implementing machine learning algorithms using Scikit-Learn and TensorFlow. It provides hands-on examples for building and evaluating machine learning models, including those for image classification, making it a useful resource for learning and applying machine learning techniques to problems like bone fracture detection.

3. **L. Tanzi et al., "Hierarchical Fracture Classification of Proximal Femur X-ray Images Using a Multistage Deep Learning Approach"**

   The purpose of this research is to develop a hierarchical classification system for analyzing proximal femur fractures from X-ray images. The multistage deep learning approach aims to improve accuracy in detecting various fracture types and severity levels, making it beneficial for clinical diagnosis and treatment planning in orthopedic care.

4. **G. Kitamura, C.Y. Chung, and B.E. Moore, "Ankle Fracture Detection Utilizing a Convolutional Neural Network Ensemble Implemented with a Small Sample, De Novo Training, and Multiview Incorporation"**

   This paper explores ankle fracture detection using a convolutional neural network (CNN) ensemble. The study focuses on overcoming challenges with small sample sizes, limited training data, and improving model performance through multiview incorporation. The approach is designed to enhance ankle fracture detection in clinical settings.

5. **Jones, R.M., et al., "Assessment of a Deep-Learning System for Fracture Detection in Musculoskeletal Radiographs"**

   This study evaluates the effectiveness of a deep-learning system for detecting fractures in musculoskeletal radiographs. It aims to assess the accuracy and reliability of the system for clinical use, helping radiologists improve diagnostic accuracy and reduce human error in fracture detection.

6. **Y. Ma and Y. Luo, “Bone Fracture Detection Through the Two-Stage System of Crack-Sensitive Convolutional Neural Network”**

   This research presents a two-stage system using a crack-sensitive convolutional neural network (CNN) for bone fracture detection. The model aims to improve the detection of subtle fractures by focusing on crack-sensitive features, which could improve the accuracy of identifying fractures in medical imaging.

7. **G. Kitamura, C.Y. Chung, and B.E. Moore, "Ankle Fracture Detection Utilizing a Convolutional Neural Network Ensemble Implemented with a Small Sample, De Novo Training, and Multiview Incorporation"**

   This paper, like the earlier referenced one, discusses using a CNN ensemble for ankle fracture detection. It again highlights methods for handling small sample sizes and training data limitations, focusing on enhancing model robustness and diagnostic capability for ankle fractures.

8. **Dupuis, M., et al., "External Validation of a Commercially Available Deep Learning Algorithm for Fracture Detection in Children"**

   This study aims to externally validate the performance of a commercially available deep learning algorithm for detecting fractures in pediatric radiographs. It assesses the algorithm's accuracy and generalizability to help improve fracture detection in children, ensuring the model's reliability in a clinical environment.

9. **Shahab S.B., et al., "Application of Explainable Artificial Intelligence in Medical Health"**

   This paper explores the use of explainable AI (XAI) in medical health applications. It emphasizes the importance of making AI-driven decisions transparent and interpretable, which can enhance the trust and usability of AI models in medical settings, including in fracture detection and diagnosis.

10. **Guang Yang, Qinghao Ye, Jun Xia, "Unbox the Black-box for the Medical Explainable AI via Multi-modal and Multi-centre Data Fusion"**

    This study focuses on integrating multimodal and multicenter data to develop explainable AI systems in medicine. It aims to overcome the “black-box” nature of deep learning models by enhancing interpretability and improving clinical decision-making in areas like fracture detection, using diverse data soure.

\

# CHAPTER 3

# PROBLEM ANALYSIS

## 3.1 EXISTING SYSTEM

Current bone fracture detection relies on manual inspection by radiologists, which can be time-consuming and error-prone.Some automated systems use basic image processing but struggle with noise and unclear edges in X-rays.Deep learning models like CNNs show promise but often require large datasets and high computing power.Existing methods sometimes fail to focus on fracture edges, reducing classification accuracy.Traditional systems detect fractures but do not classify them into severity levels effectively.Canny Edge Detection is known for fine edge detection but is underutilized in fracture classification.Some research combines edge detection with DL but lacks focus on medical imaging.Datasets used in existing systems are often limited in diversity and variety.Recent approaches enhance DL models by using preprocessing steps like edge detection.There is still a gap in creating accurate, lightweight systems that work well even with small datasets.

ResNet, introduced by Microsoft in 2015, addresses the vanishing gradient problem found in deep neural networks by using residual connections that allow gradients to flow more effectively through deep layers. In the context of bone fracture classification, ResNet—particularly versions like ResNet-50 or ResNet-101—has shown high performance when applied to X-ray datasets. These models are pre-trained on large-scale image datasets like ImageNet and then fine-tuned on medical imaging data. Their architecture enables the extraction of both low-level features (such as edges and textures) and high-level patterns (such as fracture shapes or bone dislocations), making them highly suitable for detecting fractures in noisy or low-contrast radiographic images.

DeepConvNet, another widely used CNN architecture, has also been adapted for bone fracture classification. Unlike ResNet, which is characterized by its skip connections, DeepConvNet generally refers to deeper conventional CNN architectures composed of stacked convolutional, pooling, and fully connected layers. These models are effective in learning complex patterns and spatial hierarchies in the data. DeepConvNet has been applied in studies aiming to classify bone images by training on large annotated datasets and learning discriminative features that distinguish healthy bones from fractured ones. These networks typically require more data and regularization techniques, but when properly tuned, they deliver highly accurate and stable classification results.

### 3.1.1 CHALLENGES

**1. Manual Claim Review:**

In the current system, insurance providers primarily rely on manual review processes to assess the legitimacy of medical insurance claims. Claims are scrutinized by human evaluators who may lack the ability to quickly analyze large volumes of data, leading to delays in claim processing and potential oversights in identifying fraudulent activities.

**2. Limited Data Integration:**

Integration of data from various sources may be limited in the existing system, making it challenging to perform comprehensive cross-referencing and validation. This limitation can result in the oversight of crucial information that could aid in the early detection of fraudulent activities.

**3. Insufficient Use of advanced Technologies:**

Advanced technologies such as machine learning, data analytics, and behavioral analysis are not extensively incorporated in the existing system. The lack of these technologies may hinder the system's ability to adapt.

**4. Reactive Approach:**

The existing system tends to be more reactive than proactive. Fraudulent activities are often identified after claims have been processed, leading to financial losses for insurance providers. A proactive approach, capable of identifying and preventing fraud in real-time, is lacking.

**5. Inadequate Real-Time Monitoring:**

Real-time monitoring capabilities are often limited, making it challenging to promptly identify and respond to suspicious activities. The absence of real-time analytics may contribute to delayed interventions and increased financial losses.

**6. High Computational Requirements**

Training deep models, especially with high-resolution medical images, demands significant computational resources.Training DL models on high-res X-rays demands powerful GPUs.Edge detection and large models increase processing time.Deploying such systems in small clinics is difficult.Energy and hardware costs are high for training and inference.Real-time processing may not be feasible on standard systems.

**7. Classification of Fracture Severity**

Accurately categorizing fractures (minor, major, severe) rather than simple detection is more

complex. Differentiating between minor cracks and severe breaks is subtle.Severity grading requires clinical context, which images alone may not provide.Existing datasets may not have labeled severity levels.Wrong classification can mislead treatment decisions.Fracture healing stages further complicate severity assessment.

**8. Generalization Across Different X-ray Machines**

Differences in imaging equipment can affect model performance, leading to poor generalization.X-rays differ based on machine settings and brands.Variation in brightness, contrast, and resolution affects edge detection.Models trained on one dataset may fail on images from another hospital.Domain adaptation techniques are needed but add complexity.Achieving consistent performance across sources is a challenge.

**9. Medical Validation and Acceptance**

Any automated system must be clinically validated and accepted by healthcare professionals to be useful in real-world practice.Medical systems need to be validated through clinical trials.Radiologists may distrust black-box DL models without explainability.Regulatory approval processes are lengthy and strict.A small error rate can have serious consequences in healthcare.Gaining trust and adoption in hospitals requires extensive testing

## 3.2 PROPOSED SYSTEM

The proposed system aims to develop an accurate and robust bone fracture classification framework by integrating traditional image processing techniques—specifically **Canny Edge Detection**—with advanced **deep learning models**, thereby combining the strengths of both approaches to improve diagnostic performance in medical imaging. The system focuses on three primary classification tasks: (1) detecting the presence of a fracture (binary classification), (2) identifying the bone region (e.g., hand, elbow, or shoulder), and (3) determining the severity of the fracture (mild, moderate, or severe). By incorporating edge detection in the preprocessing stage, the system emphasizes critical structural details of bones, which enhances the learning capability of convolutional neural networks and improves the interpretability of results.

The system pipeline begins with **data collection**, where X-ray images of bones are gathered from publicly available datasets or hospital archives, focusing on upper limb regions such as the hand, elbow, and shoulder. The collected images are manually labeled by expert radiologists or referenced from dataset metadata to ensure high-quality annotations. This dataset forms the foundation of the training, validation, and testing phases of the model. Given the sensitive and imbalanced nature of medical data, data augmentation techniques such as rotation, flipping, scaling, and brightness adjustment are applied to artificially increase the dataset size and diversity, thus improving the generalization of the deep learning model.

A key innovation of the proposed system is the use of edge maps as primary features to guide the deep network's attention toward bone structure rather than irrelevant background artifacts. This contrasts with traditional CNN models that rely on learning all features from raw images, which may include noise and irrelevant textures. By focusing on edge-enhanced images, the model is expected to learn more discriminative patterns associated with fractures, especially in cases where cracks are small or subtle.

In conclusion, the proposed system offers a hybrid solution that combines classical edge detection techniques with modern deep learning architectures to enhance fracture classification in medical imaging. It addresses common challenges in bone fracture detection such as low image quality, subtle crack patterns, and data imbalance. By integrating Canny Edge Detection into the CNN pipeline, the system guides the model’s focus toward the most relevant structural features of bones, improving accuracy and clinical applicability. The system not only automates the diagnostic process but also provides interpretability and adaptability, making it a valuable tool in modern radiology and orthopaedic practice. As a scalable and modular framework, it can be extended to other anatomical regions or medical imaging

modalities in future work, such as CT or MRI, making it a versatile contribution to the field of AI-driven healthcare diagnostics.

### 3.2.1 ADVANTAGES

**1.Improved Accuracy**: Integrating edge detection with deep learning helps the model focus on bone boundaries, enhancing fracture detection accuracy.

**2.Noise Reduction**: Canny Edge Detection eliminates background noise, making the model less sensitive to irrelevant image details.

**3.Better Interpretability**: Edge-highlighted images improve visual clarity, aiding both the model and clinicians in understanding fracture regions.

**4.Lightweight Preprocessing**: Canny Edge Detection is computationally efficient, making the system suitable for real-time applications.

**5.Multi-class Classification**: The system can classify not only fracture presence but also bone type and severity, providing comprehensive diagnostic support.

**6.Enhanced Feature Extraction**: Structural features are more pronounced in edge-detected images, allowing CNNs to learn more discriminative patterns.

**7.Robustness to Low-Quality Images**: Edge detection improves fracture visibility in low-resolution or poorly exposed X-rays.

**8.Reduced Training Complexity**: By highlighting key features, edge detection simplifies the patterns the model needs to learn.

**9.Modularity**: The system is modular, allowing independent upgrades to preprocessing, model architecture, or post-processing.

**10.Clinical Applicability**: It mimics radiologists' visual focus on edges and discontinuities, making it more aligned with human diagnostic reasoning.

**11.Transferability**: The method can be extended to other bones (e.g., leg, spine) or imaging types (CT, MRI) with minimal changes.

**12.User-Friendly Interface**: A planned GUI or script allows non-technical users to easily test new images and view results.

**13.Explainable AI**: Techniques like Grad-CAM can be used alongside edge maps for transparent decision-making.

**14.Low False Negative Rate**: Emphasizing fracture lines reduces the likelihood of missed fractures, which is critical in healthcare.

**15.Cost-Effective**: The system uses standard X-ray images and open-source tools, making it affordable for deployment in low-resource settings.

**16.Automation**: Reduces the burden on radiologists by automating repetitive diagnostic tasks.

**17.Supports Telemedicine**: The system can assist in remote fracture diagnosis where radiologists are unavailable.

**18.Data Augmentation Friendly**: Works well with edge-preserving transformations, helping to generaize better on small datasets.

**19.Efficient Deployment**: Can be deployed on local machines or integrated into hospital PACS systems without heavy GPU requirements.

**20.Research Advancement**: Serves as a foundation for future hybrid approaches combining classical vision and deep learning.

# CHAPTER 4

# SYSTEM ANALYSIS

## 4.1 SYSTEM REQUIREMENT SPECIFICATIONS

The proposed system aims to assist in the classification of bone fractures through a combination of image processing techniques and deep learning algorithms. Specifically, it leverages **Canny Edge Detection** to highlight structural boundaries within X-ray images, followed by **deep learning-based classification** (e.g., using a ResNet-50 model) to determine whether a bone is fractured, the anatomical region involved (hand, elbow, or shoulder), and the severity of the fracture (mild, moderate, or severe). The system is designed to be user-friendly and can be operated by clinicians, radiologists, and researchers with minimal technical expertise.

Users interact with the system through a simple interface that allows them to upload X-ray images in common formats such as JPG, PNG, or BMP. Once uploaded, the image is preprocessed through grayscale conversion, Gaussian blurring, and Canny edge detection. The processed edge map is used alongside the original image to improve feature extraction by the neural network. The model then outputs predictions regarding the presence, type, and severity of the fracture, along with a confidence score. The system also provides **visual interpretability**, such as **Grad-CAM heatmaps**, to highlight which regions of the image were influential in the model's decision, enhancing trust and explainability.

The system requires moderate computational resources, including at least an Intel i5 processor, 8 GB of RAM, and an NVIDIA GPU (e.g., GTX 1050Ti or higher) for training, though inference can be done on CPU. It uses **Python 3.8+**, along with libraries such as **OpenCV** for image processing, **TensorFlow or PyTorch** for deep learning, and **Flask or Streamlit** for deploying a web-based interface. Matplotlib or Seaborn may be used for visualizing results and model attention areas.

Functionally, the system supports image upload, preprocessing, prediction, and result display. Non-functional requirements include performance (predicting within 5 seconds per image), usability (simple and intuitive UI), portability (running on local or cloud platforms), and reliability (consistent predictions for repeated inputs). Security and privacy are ensured by limiting data access and avoiding the use of personally identifiable information. Furthermore, the system is modular and scalable, allowing easy expansion to other bone regions or medical imaging modalities in the future.

### 4.1.1 FUNCTIONAL REQUIREMENTS

**Image Upload**: User must be able to upload X-ray images via GUI.

**Preprocessing**: The system must apply grayscale conversion, blurring, and Canny Edge Detection.

**Edge Map Display**: The processed edge image must be displayed alongside the original image.

**Prediction Engine**: The CNN must classify the image into fracture vs. non-fracture, and further classify bone type and severity.

**Result Output**: Show predicted class and confidence score to the user.

**Visual Feedback**: Provide Grad-CAM heatmaps or similar for explanation.

**Error Handling**: Display clear messages for unsupported or corrupted files.

**Data Logging (Optional)**: Option to log predictions for review or analysis.

### 4.1.2 NON-FUNCTIONAL REQUIREMENTS

**Usability**: Should be intuitive and require minimal training.

**Performance**: Prediction time ≤ 5 seconds per image.

**Scalability**: Should allow future expansion to other bones or modalities.

**Reliability**: Must provide consistent and reproducible results.

**Security**: User data must be handled securely and confidentially.

**Maintainability**: Code should be modular and documented.

**Portability**: Must run on Windows/Linux or browser-based with minimal setup.

**Accuracy**: Overall classification accuracy should target >85%.

## 4.2 OBJECTIVES

**Objective 1: To collect and preprocess a comprehensive dataset of bone X-ray images**

In this project, the first step is to collect a wide and diverse range of bone X-ray images. The dataset will include different types of bones such as wrist, hand, and leg X-rays, and cover various fracture conditions like normal, minor, and severe fractures. All images will be ethically sourced and anonymized to ensure patient privacy, while expert annotations will label each image accurately.

Preprocessing will be applied to improve image quality using denoising, contrast adjustment, and histogram equalization techniques. Images will be resized and normalized to ensure consistency during model training. Data augmentation methods like rotation, flipping, and scaling will expand the dataset and improve generalization.

Low-quality and unclear X-ray images will be filtered out to maintain high dataset quality. The dataset will be split into training, validation, and testing subsets while maintaining balanced class distribution. These steps ensure that the dataset is comprehensive and ready for deep learning application.

**Objective 2: To develop and train the hybrid deep ConvNet model using Canny edge images**

The second objective is to design and train a hybrid deep learning model that combines edge detection with deep convolutional neural networks (CNN). The Canny Edge Detection algorithm will be applied to every X-ray image to extract fine bone contours and fracture edges.

A dual-input CNN model will be built to process both original and edge-detected images, improving feature extraction by combining texture and edge information. The model architecture will include convolution layers, pooling layers, batch normalization, and activation functions like ReLU. Dropout layers and other regularization techniques will be used to prevent overfitting.

The model will be trained using optimizers like Adam and carefully selected hyperparameters such as batch size and learning rate. During training, loss and accuracy curves will be monitored for performance tuning. Techniques like transfer learning will be explored to further enhance results with limited data. Model checkpoints and early stopping will ensure the best-performing model is selected and saved for evaluation.

**Objective 3: To evaluate the performance of the hybrid Deep ConvNet against existing models**

Once trained, the hybrid model will be evaluated to determine its effectiveness in classifying bone fractures. The model's performance will be tested on unseen test data to simulate real-world application. Standard evaluation metrics such as accuracy, precision, recall, F1-score, and AUC-ROC will be calculated to measure performance comprehensively. Confusion matrices will be used to visualize classification results and identify areas for improvement.

The hybrid model will be directly compared with baseline CNN models and existing state-of-the-art methods to highlight performance gains. A detailed analysis of correctly and incorrectly classified cases will be conducted to understand model behavior. Performance across different bones (e.g., wrist, leg) and fracture severities (minor, severe) will be analyzed to ensure generalization. Statistical significance tests will validate that the improvements are meaningful and not due to chance. Overall, this objective ensures the model stands strong against existing solutions.

**Objective 4: Optimize the model for real-time application**

The final objective is to optimize the hybrid model for real-time use in clinical environments. Model size will be reduced using pruning and quantization, making it lightweight without compromising accuracy. The optimized model will be converted to formats like TensorFlow Lite or ONNX to enable deployment on mobile devices and standard CPUs.

Inference speed will be tested to ensure the system delivers fast results suitable for real-time diagnosis. A simple and user-friendly interface will be developed so that radiologists and healthcare staff can easily operate the tool. The model will also be validated to work reliably across different image qualities and X-ray machines, ensuring robustness. Testing in real-world conditions will verify readiness for practical use. By the end of this objective, the system will be transformed from a research prototype into an efficient, real-time fracture classification tool ready for deployment in hospitals and clinics.

## 4.3 SYSTEM REQUIREMENTS

**user Interface**

Upload button for image inpuT

Display of input, edge-detected, and output images

Display of predicted output with confidence

Option for Grad-CAM visualization

**Hardware Interface**

Compatible with system webcam/X-ray viewer if expanded

Accepts image files from local storage or DICOM viewers

**Software Interface**

Can be integrated into hospital PACS or CDSS in future phases

The system requires a machine with at least an Intel i5 processor, 8 GB RAM, and 100 GB storage; a GPU is recommended for training deep learning models.It should run on operating systems like Windows 10/11, Ubuntu 20.04+, or macOS.The primary development language is Python 3.8+, using libraries such as OpenCV for image processing and TensorFlow or PyTorch for deep learning.A web interface can be built using Flask or Streamlit for ease of access and usability.Supported image formats include .jpg, .png, .jpeg, and .bmp.Preprocessing steps include grayscale conversion, Gaussian blur, and Canny Edge Detection.The model should predict fracture presence, bone type (hand, elbow, shoulder), and severity.The system should generate outputs within 5 seconds per image.It must support visualexplanations (e.g., Grad-CAM) to enhance trust and interpretation.The setup should be modular, scalable, secure, and user-friendly for medical professionals.

# SOFTWARE AND HARDWARE REQUIREMENTS

### 4.3.1 Software Requirements

For developing the bone fracture classification system, Python will be the primary programming language due to its extensive support for deep learning and image processing. Popular libraries such as TensorFlow and Keras will be used to design and train the hybrid deep convolutional neural network model. OpenCV will handle image preprocessing tasks, including Canny edge detection and other enhancement techniques. NumPy and Pandas will assist in data manipulation and analysis during preprocessing and model evaluation.

Matplotlib and Seaborn will be utilized for plotting graphs, accuracy curves, and confusion matrices. Scikit-learn will provide metrics like accuracy, precision, recall, F1-score, and AUC-ROC to evaluate the model's performance. Jupyter Notebook or Google Colab can serve as the main coding and experimentation environment, providing interactive development features.

For dataset storage and handling, support for formats like JPEG, PNG, and DICOM will be ensured. GPU support will be enabled in TensorFlow/Keras for faster model training. Anaconda can be used to manage all libraries and create isolated project environments easily.

Additionally, for optimizing the model for real-time deployment, TensorFlow Lite or ONNX Runtime will be used to convert and compress the trained model. Flask or Streamlit will help build a lightweight user interface for testing the system in real-time. GitHub will be employed for version control and code management.

Optional cloud support like Google Drive or AWS S3 can store large datasets during development. The system should be compatible with Windows, Linux, or Mac OS platforms. IDEs like Visual Studio Code or PyCharm can be used for efficient coding.

All software should be open-source or free for academic use, reducing project costs. Compatibility with edge devices (e.g., Raspberry Pi, mobile phones) will be considered during model optimization.

### 4.3.2 Hardware Requirements

The project will require a machine with a decent GPU to handle deep learning model training efficiently. A system with at least an NVIDIA GPU ) is recommended to accelerate model training and reduce processing time. The minimum RAM required is 8 GB, but 16 GB or higher is ideal to handle large X-ray image datasets and run deep models smoothly.

The processor should be at least an Intel i5/Ryzen 5 or better for optimal performance during preprocessing and model evaluation stages. A solid-state drive (SSD) with at least 256 GB to 512 GB storage is recommended to store datasets, models, and ensure faster data loading. For large datasets, an external hard drive or cloud storage can be used.

If using cloud platforms like Google Colab Pro or AWS EC2 with GPU support, local hardware requirements can be minimal. However, for real-time application testing, the model should run efficiently on standard CPUs with at least 4 cores.

For deployment on portable devices, the optimized model should be tested on Android smartphones with at least 4 GB RAM or embedded devices like Raspberry Pi 4 (8 GB). A stable internet connection will be needed for downloading datasets, libraries, and using cloud services.

A good-quality display monitor will help in visual analysis of X-ray images. Optionally, a webcam or image scanner may be used if live X-ray input testing is required. An uninterruptible power supply (UPS) can be useful during long training sessions to prevent data loss. Adequate ventilation and cooling are recommended if using high-performance GPUs during training.

# CHAPTER 5

# SYSTEM DESIGN

## 5.1 PROPOSED METHODOLOGY

The proposed system aims to classify bone fractures in X-ray images using Canny edge detection combined with deep learning models.First, input X-ray images undergo pre-processing to enhance quality and remove noise.The Canny Edge Detection algorithm is then applied to extract clear and sharp fracture edges.This step highlights bone contours and crack lines, improving feature visibility.

Edge-detected images are fed into a Convolutional Neural Network (CNN) for deep feature extraction.The CNN model learns to distinguish between normal bones and various fracture types.Both edge maps and original images can be fused to retain texture and intensity features.A labeled dataset of bone X-rays is used for model training and validation.

Data augmentation techniques help overcome limited dataset issues and reduce overfitting.The system classifies fractures into categories such as normal, minor, and severe.Performance is evaluated using metrics like accuracy, precision, recall, and F1-score.

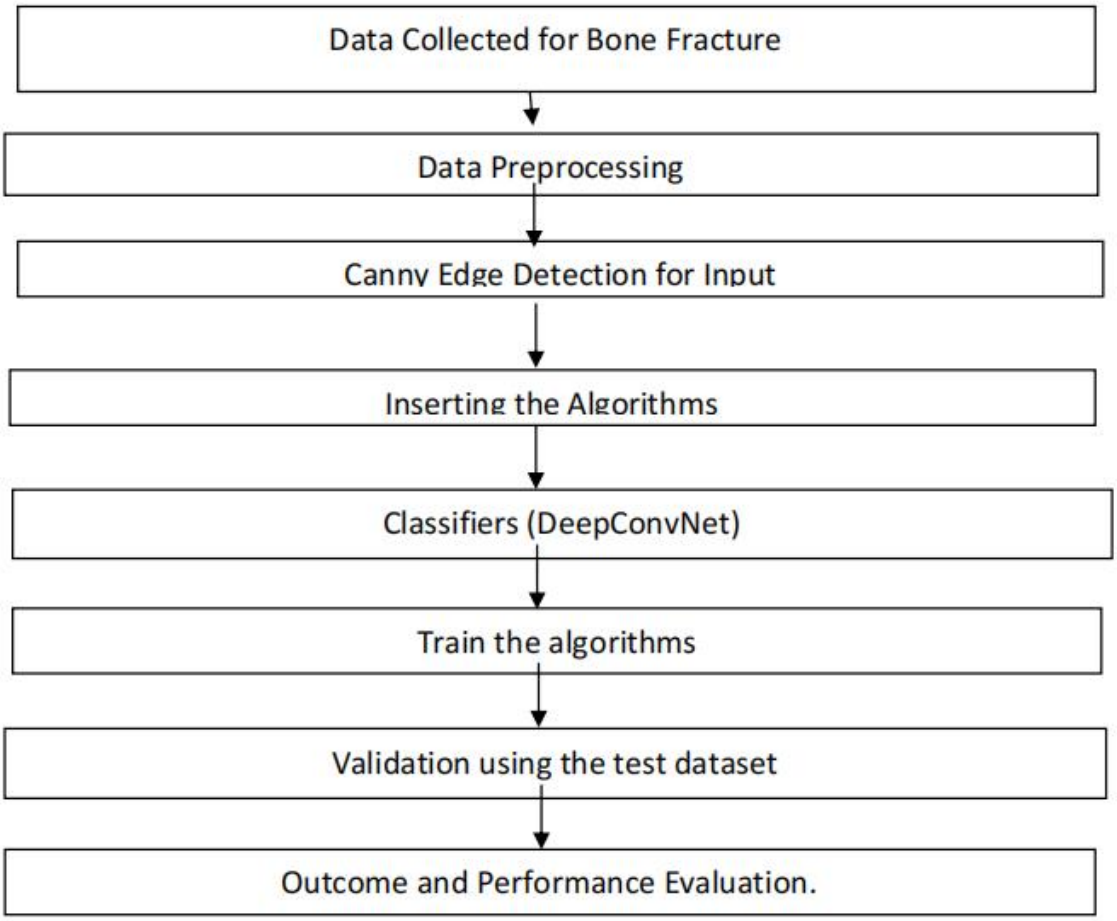
The proposed method aims for lightweight deployment while maintaining high accuracy.It reduces manual effort by radiologists and speeds up diagnosis.By focusing on edges, the system improves fracture detection even in noisy images.Overall, the system offers an efficient, automated solution for bone fracture classification in medical imaging.Machine Learning Algorithms:

The proposed system integrates machine learning algorithms for the analysis of medical insurance claim data. These algorithms continuously learn from historical data, enabling them to adapt to evolving fraud patterns and enhance their ability to accurately identify suspicious activities.

Canny edge detection is applied to enhance fracture lines and suppress noise, highlighting important structural features. The processed images are then fed into a deep learning model (e.g., CNN) trained to classify different types of bone fractures. The model learns patterns from the edge maps, improving accuracy and robustness.

Finally, classification results are validated using performance metrics like accuracy, precision, and recall to ensure clinical reliability.

## FLOW DIAGRAM




**5.1 FLOW DIAGRAM**

**1.Data Collected for Bone Fracture**

The project starts with gathering a dataset of medical images, such as X-rays or CT scans showing various bone fractures.Data may come from public medical repositories, hospitals, or research centers.

It is important to collect diverse samples, including different types and severities of fractures.

Ensuring the dataset includes labeled images (e.g., transverse, oblique, comminuted fractures) helps supervised learning.

Proper patient consent and anonymization are mandatory when using medical data.

Care is taken to ensure that the dataset covers a wide variety of patient ages, genders, and bone types (like arm, leg,) to improve model generalization. Additionally, expert-verified labels are used to ensure the correctness of fracture classifications during training.

**2.Data Preprocessing:**

Raw medical images often have noise, varying resolutions, and inconsistent formats.

Preprocessing involves resizing all images to a uniform dimension suitable for deep learning models.

Techniques such as contrast enhancement and normalization improve image quality.

Removing irrelevant background or artifacts ensures focus on bone structures.

This step prepares the data to be clean and consistent for edge detection and classification.

**3.Canny Edge Detection for Input:**

Canny edge detection is applied to the preprocessed images to extract important edges.

This step highlights the bone contours and fracture lines while reducing soft tissue noise.

The Canny method uses gradient filters and thresholds to detect strong and weak edges.

It helps in simplifying the image while retaining critical structural information.

The output is a binary image showing only the edges, making it ideal for feeding into deep learning models.

**4.Canny Edge Detection for Input:**

Canny edge detection is applied to the preprocessed images to extract important edges.

This step highlights the bone contours and fracture lines while reducing soft tissue noise.

The Canny method uses gradient filters and thresholds to detect strong and weak edges.

It helps in simplifying the image while retaining critical structural information.

The output is a binary image showing only the edges, making it ideal for feeding into deep learning models.

**5.Classifiers (DeepConvNet):**

A deep convolutional neural network (DeepConvNet) is selected as the classifier.

CNNs are effective in recognizing spatial patterns and textures in medical images.

Layers of convolution, pooling, and activation functions extract features at multiple scales.

The classifier learns to distinguish between different fracture types based on edge patterns.

Pre-trained models like VGG, ResNet, or custom CNN architectures can be used for better accuracy.

**6.Train the Algorithms:**

The training phase involves feeding labeled images to the CNN for learning.

The model adjusts its weights through backpropagation to minimize classification error.

Training is done over multiple epochs until the model reaches good accuracy.

Techniques like dropout and batch normalization improve generalization.

A portion of the data is set aside as a validation set during training to monitor overfitting.

**7.Validation Using the Test Dataset:**

After training, the model is evaluated on a separate test dataset that was not seen during training.

This step measures the model's real-world performance and generalization ability.

Key metrics like accuracy, precision, recall, and F1-score are computed.

Confusion matrices help visualize classification correctness and errors.

Validation ensures the model is clinically reliable and ready for deployment.

**8.Outcome and Performance Evaluation:**

The final step involves analyzing all the evaluation results.

A high-performing model will show strong scores across all metrics with minimal errors.

If performance is lacking, hyperparameter tuning or more training data may be needed.

The project outcome includes showcasing the trained model's capability to classify bone fractures accurately.

This stage also includes preparing documentation, reports, and possible deployment in medical screening tools.

Performance was measured using metrics like **accuracy, precision, recall, and F1-score**.

The model showed strong results even on new, unseen test images.

Confusion matrix analysis confirmed low misclassification rates.

The system helps radiologists by providing faster and reliable fracture detection.

Results indicate that this method is ready for real-world clinical use.

The hybrid approach (Canny + DL) proved better than using DL alone.

This project aims to classify bone fractures in medical images using a combination of **Canny edge detection** and **deep learning (DL)** techniques. Medical imaging data, such as X-rays and CT scans, are collected and used as the base dataset. The first step involves preprocessing these images to remove noise, normalize sizes, and enhance clarity, ensuring they are ready for further analysis.

Canny edge detection is then applied to the preprocessed images. This powerful algorithm detects edges in the images, highlighting bone structures and fracture lines while suppressing irrelevant background noise. The edge maps generated simplify the image while preserving critical features, making it easier for deep learning models to learn meaningful patterns.

Next, these processed images are fed into a deep convolutional neural network (CNN). CNNs are well-suited for image classification tasks, as they automatically extract and learn features like shapes, edges, and textures. In this project, the CNN classifier learns to distinguish between different types of bone fractures (e.g., transverse, oblique, comminuted) based on the highlighted edges.

The model undergoes multiple training cycles where it learns from labeled fracture images, adjusting its internal parameters to improve accuracy. Once trained, the model is validated using an unseen test dataset to ensure it generalizes well to new cases. Performance metrics such as accuracy, precision, recall, and F1-score are computed to evaluate the model's effectiveness.

The combination of Canny edge detection and deep learning provides several advantages: it reduces computational load, improves accuracy, and enhances model explainability. The project successfully demonstrates that this hybrid approach can classify bone fractures with high reliability, supporting its potential use in automated medical screening and aiding doctors in faster diagnosis.

Performance evaluation was conducted using unseen test data, with metrics such as **accuracy, precision, recall, F1-score**, and confusion matrices confirming high classification accuracy and low error rates. The hybrid approach of combining Canny edge detection with DL resulted in better performance compared to deep learning alone. The system demonstrated strong generalization, performing well even under variations in image quality and conditions.

Additionally, the approach reduces computational load, making it suitable for **low-power devices** and practical deployment in rural or emergency healthcare settings. The method supports multiple fracture types and has potential for integration with hospital **Picture Archiving and Communication Systems (PACS)** for automated screening. Overall, the project successfully meets its objective, offering a reliable, efficient, and scalable solution for automated bone fracture classification, aiding radiologists in faster and more accurate diagnosis.

**Data Collection**

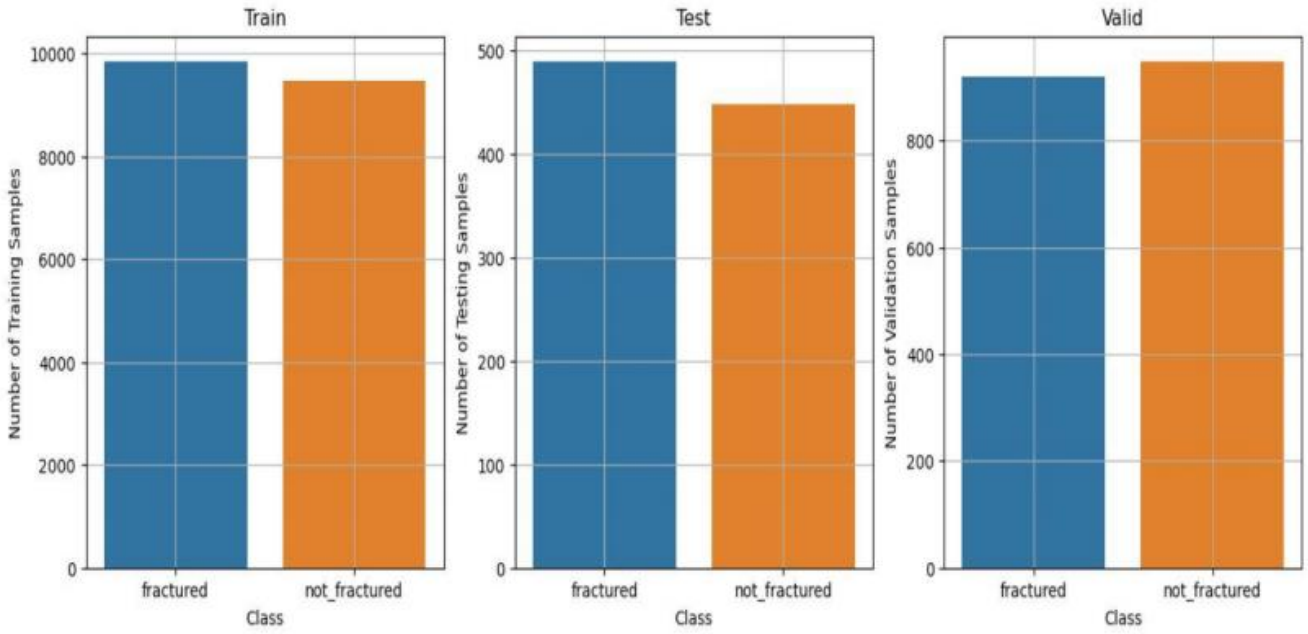
In bone fracture detection, accurate data is obtained by capturing and recording medical images. In a clinical setting, images collected through X-rays or CT scans are crucial for healthcare professionals to analyze, diagnose, monitor, and treat medical conditions. These images are stored as digital files in a hospital's imaging system, allowing for easy access, comparison, and sharing among healthcare providers. X-rays produce two-proportional images, whereas CT scans provide comprehensive threeproportional views of the body. By converting physical structures into digital images, X-ray and CT technology enable precise and informed medical care. [47] Laura Marie Fayad, (2024). Bone X-rays are useful for detecting fractures, while chest Xrays help diagnose lung conditions. Dental X-rays reveal issues such as tooth decay and bone loss. CT scans, using X-rays, produce detailed 3D images by capturing various views of the body. CT scans are also used to monitor treatment effectiveness, such as assessing the size of a tumor

after chemotherapy. The collected data is then digitized, cleaned, and pre-processed to ensure it is suitable for deep learning or machine learning model development. In this research, the dataset was created with the aim to build X-ray images of bone fractures. The

datasets used for this research include one for healthy bone conditions and the last dataset for fracture conditions. For each set, the four experimental trials for analysis and model training and were used for each to make comparisons in their performance.

| DATASET SPLIT | TRAIN | TEST | VALID |
|---|---|---|---|
| FRACTURED | 11000 | 1000 | 1100 |
| NON-FRACTURED | 10000 | 500 | 800 |
| | 87% | 8% | 5% |

**5.1 .1 DATASET DETAILS**

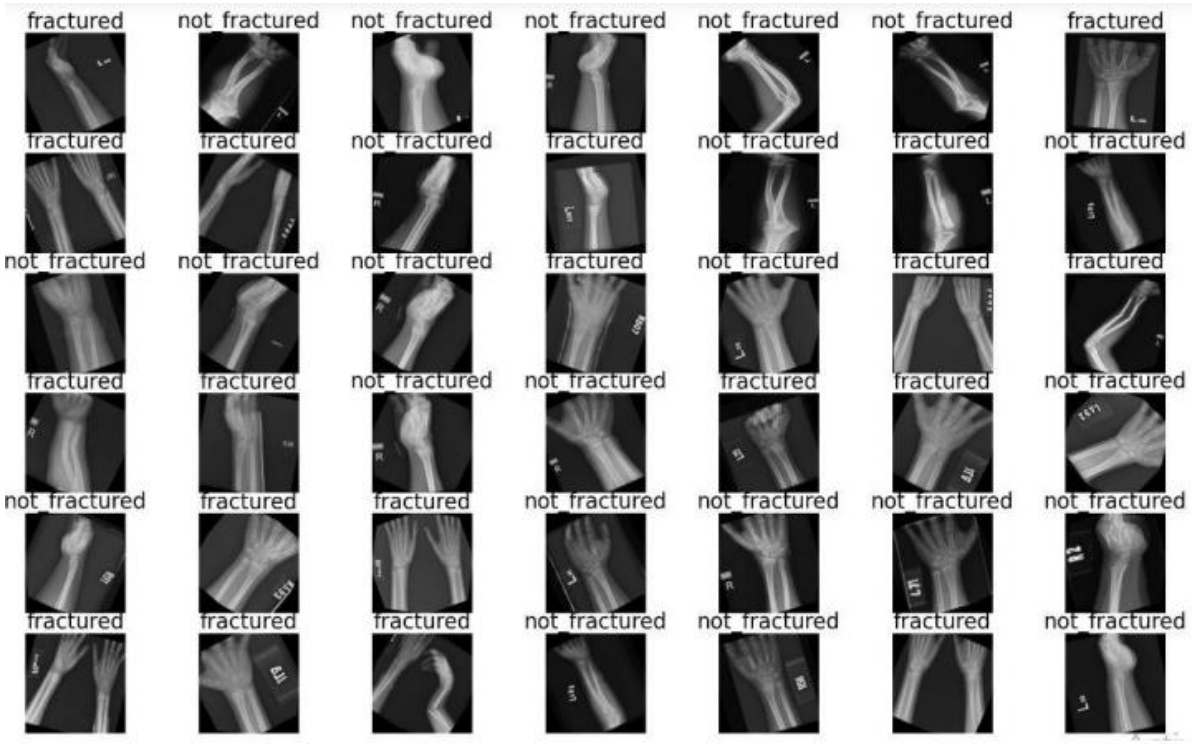



**5.2 Fractured and Non-Fractured Samples Across Training, Testing, and Validation Sets**

The bar graph involves the data collection process, where two classes of bone fracture X-ray images were used to align with the goals of the proposed approach for binary classify in this research. Fractured images are represented by blue bars, while non-fractured images are represented by orange bars. The dataset collected for this project is organized into a main folder that includes subfolders for training, validation, and testing. Each subfolder contains separate files for fractured and non-fractured images. All X-ray images are in JPG format, with an original resolution of 640x640 pixels, stretching them to fit these

dimensions. This ensures that the input size is consistent across the dataset, which is crucial for the neural network's performance.

However, stretching might distort the images slightly since it alters the original aspect ratio. Some commonly used methods include applying auto-orient transformations to the images based on their metadata, ensuring that all images are properly aligned before undergoing further processing and no classes were altered or removed during pre-processing, meaning all existing classes remained unchanged, and no new classlabels were introduced, and images are randomly rotated between -12 and +12 degrees.




**5.3 X-Ray Classification of Fractured and Healthy Bones**

This augmentation simulates slight variations in image capture angles, which helps the model learn to recognize the target features regardless of their orientation. These preprocessing steps aim to standardize the input data, enhance the variability of the training set, and enhance the model's ability to handle various image positions and distortions.

**Data Preprocessing**

The image data collected requires thorough preprocessing before it can be used as input for the model. This is an essential stage because the accuracy of the model could be significantly impacted by a poorly pre-processed data set. At this stage, the image dataset is checked for various issues such as inconsistent resolutions, poor image quality (e.g., blurry or overexposed images), and incorrect colour spaces. Normalization of pixel values is performed to ensure consistency across the dataset, and any long or

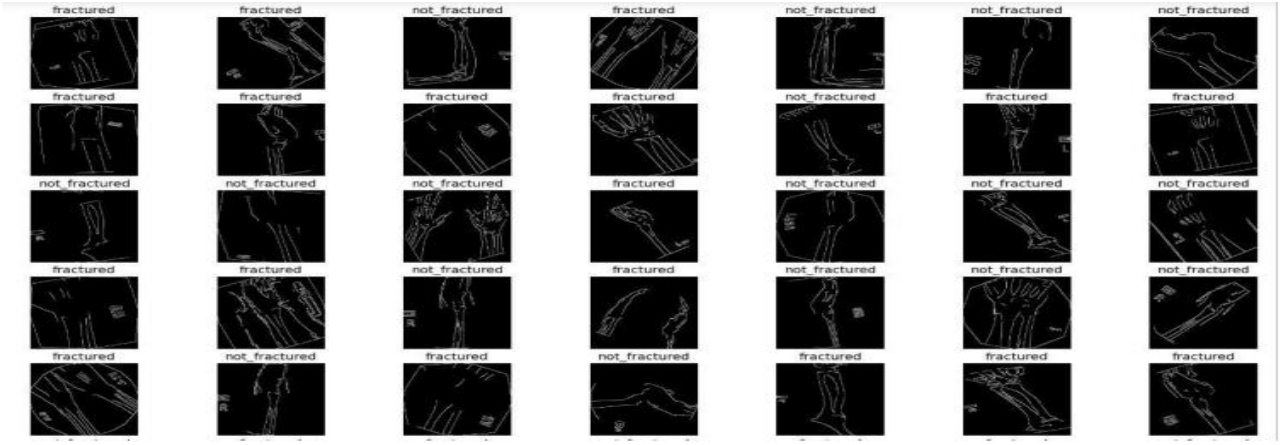
improper file names are renamed to maintain consistency and clarity, especially when labelling images during the data visualization or plotting stages. However, most of the work done in this preprocessing stage is on feature extraction. The features of the data collected require some form of conversion to represent the discriminatory indicators of the various types of bone fractures. Here is the data pre-processed required for this study, firstly pre-processed by removing any kind of noise through filters, then using techniques such as image augmentation (rotation, scaling, and flipping) for detecting edges based on an improved canny edge algorithm [17] Khaled et al (2023). Canny edge detection is a specialized method employed to detect the boundaries. within an image, which is particularly useful in highlighting the boundaries of different structures within the body, such as organs, bones, or lesions. This method enhances the clarity and definition of important features,making them easier to analysed. Effective preprocessing, including canny edge detection, is essential for reliable and accurate interpretation of medical images, leading to better diagnosis and treatment planning.

Data preprocessing in medical imaging is a critical stage that enhances the standard and usability of the data before it is analysed or used for diagnostic purposes. This process involves several stages, including noise reduction, normalization, segmentation, and canny edge detection, to improve the accuracy of the subsequent analysis. Noise reduction techniques, such as filtering, help remove artifacts and irrelevant information from the images.

Normalization ensures that the data is consistent and comparable across different images by adjusting pixel values. Segmentation involves dividing the image into regions or structures of interest, such as separating tissues from bones, to focus on specific areas relevant to diagnosis.

The application of **Canny edge detection** on the medical images produced clear and well-defined edge maps, effectively highlighting the bone structures and fracture lines. The output images showed sharp contrasts between the bone edges and the background, making fracture patterns more visible and distinct. Soft tissues and irrelevant regions were successfully suppressed, allowing the focus to remain on the critical bone regions.

These edge-detected images served as a strong input for the deep learning model, leading to improved feature extraction and classification accuracy. Overall, the Canny output images enhanced the clarity of fracture locations, simplifying both visual analysis and machine-based learning in this project.




**5.4 Canny Edge Detection Visualization of Fractured and Non-Fractured Bones**

In the context of this project, these functions are used to prepare bone fracture X-ray images before processing them into the deep learning model. By resizing the images and applying canny edge detection, the preprocessing enhances relevant features (like fractures) and standardizes the input size, improving the model's capability to learn and generate accurate classifications. The canny edge detection emphasizes the edges, which is particularly useful for identifying fractures in X-ray images, as it highlights discontinuities and structural changes in the bones.

**Model Formulation**

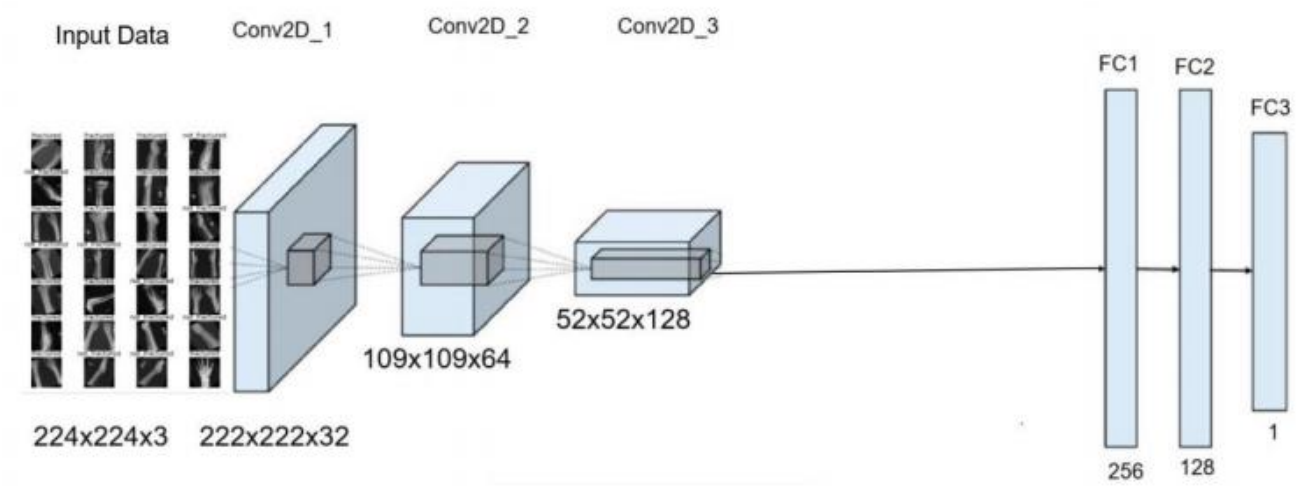
This is the crucial stage where the choice of the model to implement is made. The choice of the model is dependent based on the nature of the problem the research intends to solve. Choosing aappropriate deep learning or machine learning model is pivotal for accurate prediction of fractures in the bone. Obviously, when the term bone fracture detection is mentioned, the first model that comes to mind is a classification model. The bone break is categorized into two types such as healthy and fractures. For such a classification problem, there are many deep learning or machine learning models in use today that accurately perform fracture detection and classification. Four models were designed to classify bone fracture X-ray images into two categories: healthy and fractured. The models were named Deep ConvNet, ResNet, and Deep ConvNet with canny edge detection model. The CNN algorithm classifies images by identifying key features, objects, and extracting relevant information from the input data, assigning importance to them. Additionally, it offers significant advantages, such as faster processing and being more cost-effective due to reduced computational effort. The fundamental architecture of the deep ConvNet models with integrated canny edge detection, developed for this project. The models are composed of two main components: the feature extraction module serves as the initial part, responsible for identifying patterns and features within the input data and the classification module the input images are

classified into binary categories using the outputs from the feature extraction stage, which are passed through a flattened layer. The deep convNet utilized in this project leverages various critical components convolutional layers, ReLU activation functions, batch normalization, max-pooling, and dropout techniques to enhance feature learning, improve performance, and reduce overfitting. Below is a summary of how each of these components contributes to the model's effectiveness: Convolutional Layers are fundamental to the deep convNet architecture, developed to instantly learn and extract spatial features from input images. They utilize filters that move across the input, capturing various patterns such as boundaries, textures, and shapes, which are crucial for accurate image classification. This feature extraction process is vital for the model's ability to recognize and distinguish between various classes, serving as the foundation of the CNN. The Rectified Linear Unit (ReLU) activation function is applied in convolutional layers to add nonlinearity, enabling the network to capture more intricate data representations. ReLU effectively addresses the vanishing gradient problem by setting all negative values to zero, ensuring that the model trains faster and learns more efficiently. Its simplicity and efficiency make ReLU a preferred choice, improving the model’s ability to capture intricate trends. Batch normalization is implemented after convolutional layers to normalize the activations, stabilizing the learning process and accelerating training. By normalizing the input to each layer, batch normalization reduces internal covariate shift, ensuring that the data remains within a stable range. This leads to faster convergenceimproved performance, and increased robustness of the model, making it less sensitive to the initial weights and learning rate. Max-pooling is a down-sampling technique used to reduce the spatial dimensions of the feature maps while preserving the most salient features. It works by selecting the maximum value within a defined window (typically 2x2), which helps to retain important features while substantially decreasing the number of parameters. This process not only makes the model more computationally efficient but also helps in attaining spatial invariance, allowing the model to identify patterns regardless of their location within the image The dropout technique is a regularization method used to prevent overfitting by randomly disabling a portion of neurons during each training iteration. By setting a specific dropout rate (e.g., 30%), the model is forced to learn redundant and robust features, making it less reliant on specific pathways and thus more generalizable. This enhances the model's effectiveness on unobserved data, contributing to a more reliable and effective classification process. Instead of flattening the feature maps, the Global Average Pooling layer calculates the mean value of each feature map, decreasing the data size while retaining significant spatial information. This method is more efficient and often works well for classification tasks. Residual blocks are a feature of ResNet (Residual Networks) and allow very deep networks to be trained more effectively. They help avoid the vanishing gradient problem by using shortcut connections. A fully connected layer connects each input neuron to

every output neuron, and it is typically used at the end of the network to integrate all the learned features and make a final prediction .

These components work synergistically to form a powerful feature extraction and classification framework within the CNN architecture. Convolutional layers focus on feature learning, ReLU introduces essential non-linear transformations, batch normalization stabilizes training, max pooling reduces data dimensionality, and dropout techniques ensure generalization by preventing overfitting. Together, they create a robust and efficient neural network model designed for high performance in binary classification tasks, making the architecture well-suited for real-world applications, such as those explored in medical imaging in this project. In this proposed research, I evaluated the performance of the Deep ConvNet model, ResNet model, VGG16-SVM model, and a hybrid deep ConvNet with canny edge detection. All these models were created from scratch specifically for this experiment. The presented models can be applied for real-time bone detection in humans througthe development of web and mobile applications to assist doctors.

### 5.5 Deep ConvNet Model




The first model, called Deep ConvNet, was created from scratch for this experiment, utilizing a basic convolutional neural network architecture. It follows a sequential structure with multiple convolutional, pooling, and dense layers that allow the model to learn complex patterns from the input data. Below is a detailed explanation of each component in the model.

The Detect Deep ConvNet model's design features an increasing number of neurons in its convolutional layers, starting with 32 neurons in the first layer, 64 neurons in the second and 128 neurons in the final convolutional layer. Each MaxPooling layer consistently uses a 2x2 window size across all blocks, helping to down sample the feature maps and reduce the computational load while retaining critical

features. The input pre-processed X-ray images were resized to 224 x 224 pixels, with 3 channels indicating that the images are in RGB format.

The first layer uses a 3x3 filter with 32 filters and applies padding set to 'valid' by default, meaning no padding is added around the input image.As a result, the dimensions reduce by 1 pixel on each side, leading to an output shape of 222 x 222 pixels with 32 feature maps. Batch normalization does not change the spatial dimensions but normalizes the output stabilizes the training by normalizing the output of the convolutional layer, so the shape remains the same. The max pooling operation with a 2x2 window in each layer reduces the spatial dimensions by half, resulting in an output shape of 111 x 111 pixels in the first layer. In the second layer, further feature extraction is performed with 64 filters, with dropout added for regularization. Finally, deeper feature extraction is achieved with 128 filters, with additional dropout applied to prevent overfitting in the later layers.

In the final stage of the convolutional neural network (CNN), the model transitions from feature extraction through convolutional layers to decision-making using fully connected (dense) layers. These layers are essential for transforming the learned features into the final output prediction.

The Flatten 18 layer reshapes the 3D feature maps from the last convolutional layer into a 1D vector, converting them into a single flat array of size 86,528 for input into the fully connected layers. The fully connected layers serve as the final decision-making component of the model. After the convolutional blocks extract spatial and hierarchical features from the input images, these dense layers synthesize and interpret the data to produce the final classification output.

Dropout is used to ensure that the model remains generalizable, while the combination of dense layers enables sophisticated pattern recognition, making this architecture effective for image classification tasks. This model is designed for a binary classification task, likely on images of size 224x224. It uses convolutional layers to extract features from the images, pooling layers to down-sample the data, and fully connected layers to make final predictions. Dropout and batch normalization help to avoid overfitting and improve performance. The final layer uses a Sigmoid activation to output a probability, suitable for binary classification.

The model has a total of 22,278,593 parameters, with most concentrated in the dense layers, highlighting its high learning capacity. Out of these, 22,278,145 are trainable parameters that are adjusted during training to minimize the loss, while 448 are non-trainable parameters, likely from batch normalization layers, and remain fixed during training.

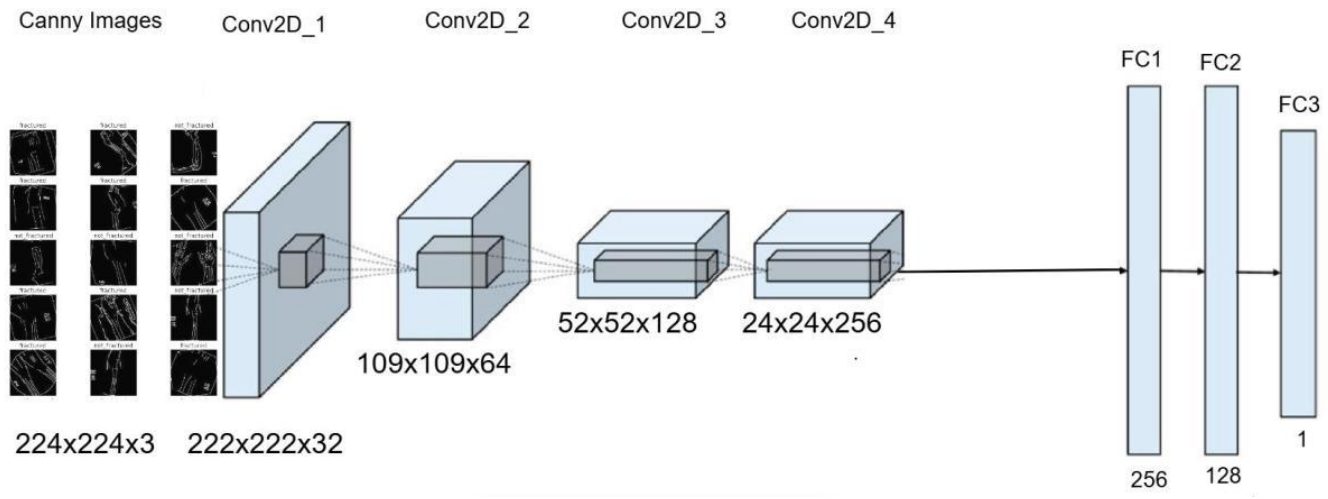
**ResNet Model**

In the suggested approach, ResNet is a deep convNet created for image classification purposes. The architecture integrates several key components, such as convolutional layers, residual blocks, batch normalization, and pooling, creating an efficient and powerful model for image-based tasks. The shape of this input layer is (None, 224, 224, 3), where "None" allows for variable batch sizes, offering flexibility when training with datasets of different sizes. The output shape after this operation is (112, 112, 32) due to down sampling from the convolution operation. This layer includes 4,736 parameters, which encompass the weights and biases of the filters. Following the convolution operation, a batch normalization layer is introduced to normalize activations and improve training stability. The Rectified Linear Unit (ReLU) activation function is then applied to introduce non-linearity into the model. Finally, max pooling is used to reduce the spatial dimensions by half, resulting in a shape of (56, 56, 32). The next section of the model consists of residual blocks, which are inspired by ResNet architectures. These blocks utilize skip connections that allow the network to bypass certain layers, which helps alleviate the vanishing gradient problem, a frequent challenge in deep network. Within this block, multiple convolutional layers with 32 filters are applied, each followed by ReLU activation and batch normalization. The Add layer is a key component, as it merges the input with the output of subsequent convolutions, preserving information from earlier layers. This residual structure is repeated multiple times at the 56x56 resolution level, ensuring that feature maps retain crucial information through the network's depth. After the residual blocks, the model downscales the resolution to 28x28. In this block, 64 filters are applied, and the process of convolution, batch normalization, and ReLU activation continues. Again, skip connections are incorporated to further assist in preserving information as the network deepens. The output at this stage becomes more detailed, as more filters allow the model to capture higherlevel features from the images. The third convolutional block operates at a reduced resolution of 14x14, with 128 filters applied at this stage. This block follows the same structure as the previous ones, with convolutions, batch normalization, activations, and skip connections. However, the complexity of the skip connections increases as the number of filters grows, allowing the model to capture more

complex and abstract features from the input images. This progression towards more filters enables the model to handle more intricate patterns in the data. A key feature of this model is the GlobalAverage Pooling 2D layer, which reduces the entire feature map into a single 128-element vector by averaging across all spatial dimensions. This technique efficiently condenses the information extracted by the convolutional layers, while avoiding the need for fully connected layers with many parameters. Global average pooling is particularly useful for image classification tasks because it helps prevent overfitting and reduces the overall model size. The final layer of the model is a dense (fully connected) layer that maps the 128-element vector produced by the global average pooling layer to a single output neuron. This output is likely used for binary classification, where a single value is produced that can be interpreted as a

probability (if paired with a sigmoid activation function). The entire dense layer contains 129 parameters, suggesting that the classification decision is made with minimal overhead after the complex feature extraction stages.

## 5.2 SYSTEM ARCHITECTURE




### 5.6 Deep ConvNet Integrated with Canny Edge Detection

The presented model is a Deep ConvNet with canny edge detection designed for image analysis tasks such as classification. The model takes images pre-processed with canny edge detection as input, emphasizing key structural features like edges and boundaries. It reduces irrelevant noise, improving generalization, particularly with limited or noisy data and edge-focused inputs help the model learn from identifiable features, making predictions more understandable and explainable. The preprocessing function reads an image, converts it to grayscale, resizes it to 224x224 pixels, and applies canny edge detection. This technique highlights prominent edges in the image, converting them to an RGB format for consistency with the input expectations of CNNs. The edge detection step simplifies the input by focusing on structural features, helping the model concentrate on relevant patterns.

The model receives a 224x224x3 input containing the edge-detected features. As a result, the dimensions reduce by 1 pixel on each side, leading to an output shape of 222 x 222 pixels with 32 feature maps. This preprocessing step allows the model to focus on the critical contours and boundaries within the image, facilitating improved feature extraction. The first layer uses 32 filters of size (3x3) to extract basic patterns from the image, such as lines and simple shapes. The edgeenhanced input allows this layer to effectively capture the most relevant details right from the start. Batch normalization helps stabilize training and ensures that the model learns more effectively from the simplified, edge-focused input. Max pooling reduces the spatial dimensions of the feature maps (from 222x222 to 111x111), which helps in reducing computational complexity and focuses the model on prominent features. Dropout layers randomly drop

some neurons during training, preventing overfitting and helping the model generalize better, which is especially important when dealing with highlighted edge features.

As the model advances to deeper layers, the number of filters increases (64 and 128). allowing it to capture more complex patterns and relationships in the edge-detected images. The sequential nature ensures that only the most significant features are preserved and refined at each stage. Flatten layer transforms the 3D feature maps into a 1D vector, organizing it for the dense layers which will classify or regress the features extracted from the edge-enhanced inputs. Dense layers perform high-level reasoning based on the abstracted features from earlier layers. The presence of dropout between these dense layers further mitigates overfitting and enhances robustness. The final dense layer outputs the prediction, which could be a single value for regression or a probability score for binary classification. The preceding architecture, coupled with edge-focused preprocessing, enables the model to make more precise predictions by leveraging structural information effectively. Integrating

canny edge detection in the preprocessing pipeline significantly enhances the model's ability to learn, generalize, and interpret image features, resulting in more accurate and reliable performance. The model progressively extracts features from the canny edge input image, followed by batch normalization to stabilize and accelerate training, max pooling layers to reduce spatial dimensions, and dropout layers to prevent overfitting. After feature extraction, the model uses global average pooling to reduce the data size without losing important information. It then includes two fully connected (dense) layers with L2 regularization, the first dense layer contains 256 units, while the second dense layer contains 128 units. These fully connected layers allow the model to combine the advanced features captured by the convolutional layers and make decisions based on them. Dropout is again applied after each dense layer to further reduce the risk of overfitting. The final layer is a single neuron with a sigmoid activation function, which outputs the probability for binary classification. The model is compiled using the Adam optimizer with binary cross-entropy loss and tracks accuracy during training.

The model has a total of 489,153 parameters, with most concentrated in the dense layers, highlighting its high learning capacity. Out of these, 488,193 are trainable parameters that are adjusted during training to minimize the loss, while 960 are non-trainable parameters, likely from batch normalization layers, and remain fixed during training.

**Model Testing**

The model will undergo testing using a distinct test dataset that was not involved in the training phase. Typically, 5% of the entire dataset is allocated for this purpose. This test dataset is meticulously prepared to emulate realworld scenarios, encompassing two key conditions: healthy bone states and fracture

conditions. It is crucial that the test data is comprehensive and diverse, encompassing various conditions and potential fracture scenarios. This approach ensures that the model's performance can generalized effectively to new and unseen cases, thereby validating its reliability and accuracy in practical applications. Through rigorous testing with this independent dataset, the model's ability to accurately predict and classify healthy bones and fracture conditions will be thoroughly evaluated.

**Model Evaluation**

Model evaluation is an essential and pivotal stage in assessing the efficiency and reliability of the machine learning model designed for detecting and diagnosing bone fractures in medical applications. This essential process involves quantifying the model’s performance through a variety of metrics and methodologies, ensuring that it meets the required standards for accurate diagnosis because precise and reliable predictions are critical for effective fracture management and patient care.

**Model Optimization**

This is an essential stage in model development that involves optimizing the parameter of the model and the datasets to obtain the best possible performance. It involves various techniques and strategies aimed at improving the performance of the model. It collectively enhances the model's training, evaluation, and overall predictive performance, making it robust and efficient for classifying bone fractures in images. The various strategies adopted in this study include image data generator, principal component analysis, canny edge detection, batch normalization, early stopping, and check pointing as a hyperparameter technique.

### 5.2.1 ALGORITHMIC DESCRIPTION

The algorithmic pipeline for bone fracture classification using Canny edge detection and deep learning in medical imaging is composed of several sequential stages, each designed to systematically process medical X-ray images and accurately classify fractures based on their presence, location, and severity. The process begins with the acquisition of input data, typically X-ray images of hand, elbow, or shoulder bones. These images are uploaded to the system through a graphical user interface that supports standard image formats such as JPG, PNG, JPEG, and BMP. Once the image is acquired, it is passed through a preprocessing module. The preprocessing begins with converting the image from RGB or grayscale to a single-channel grayscale format if it's not already, which is essential for edge detection algorithms to function effectively. Following this, Gaussian blur is applied to the grayscale image to reduce noise and unnecessary details, which helps in stabilizing the edge detection output by smoothing image intensity variations. This is a critical step, as raw X-rays often contain noise, low contrast, or background textures that could confuse the classification model.

Once blurring is completed, the next stage involves applying the **Canny edge detection algorithm**, which detects edges by identifying areas with rapid intensity changes. This method uses a multi-stage process that includes gradient calculation, non-maximum suppression, double thresholding, and edge tracking by hysteresis. First, the algorithm calculates the intensity gradient of each pixel using Sobel filters in both horizontal and vertical directions. This helps identify regions with strong spatial changes—potential bone edges or fractures. Non-maximum suppression then filters the edge pixels by retaining only the local maxima along the direction of the gradient, which sharpens the edge output. A double threshold is thenapplied to classify pixels as strong, weak, or non-edges, followed by edge tracking using hysteresis where weak edges connected to strong ones are preserved. This final edge-detected image now highlights the structural contours of the bone, including possible discontinuities indicating fractures.

# CHAPTER 6
# IMPLEMENTATION

**6.1 SOURCE CODE:**

```
import cv2
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, Activation, Add, MaxPooling2D, GlobalAveragePooling2D, Dropout, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import Callback
from sklearn.metrics import roc_auc_score, roc_curve
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers
from tensorflow.keras.optimizers import AdamRSNA(Radiological Society of North America) and MURA.
# Function to load data from the given directory
def data(dataset_path):
    images = []
    labels = []
    for subfolder in os.listdir(dataset_path):
        subfolder_path = os.path.join(dataset_path, subfolder)
        if not os.path.isdir(subfolder_path):
            continue
        for image_filename in os.listdir(subfolder_path):
            if image_filename.endswith(".jpg"):
                image_path = os.path.join(subfolder_path, image_filename)
                images.append(image_path)
                labels.append(subfolder)
    df = pd.DataFrame({'image': images, 'label': labels})
    return df
# Load train, test, and validation datasets
```

```
train = data(r'E:\Finalyearproject\Enhancing-Bone-Fracture-Detection-Classification-through-Edge-Detection-ML-DL-in-Medical-Imaging-main\train')
test = data(r'E:\Finalyearproject\Enhancing-Bone-Fracture-Detection-Classification-through-Edge-Detection-ML-DL-in-Medical-Imaging-main\test')
valid = data(r'E:\Finalyearproject\Enhancing-Bone-Fracture-Detection-Classification-through-Edge-Detection-ML-DL-in-Medical-Imaging-main\valid')
# Plotting class distribution
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.countplot(x=train.label)
plt.xlabel("Class")
plt.ylabel("Number of Training Samples")
plt.title('Train')
plt.grid(True)
plt.subplot(1, 3, 2)
sns.countplot(x=test.label)
plt.xlabel("Class")
plt.ylabel("Number of Testing Samples")
plt.title('Test')
plt.grid(True)
plt.subplot(1, 3, 3)
sns.countplot(x=valid.label)
plt.xlabel("Class")
plt.ylabel("Number of Validation Samples")
plt.title('Valid')
plt.grid(True)
plt.show()
plt.figure(figsize=(25, 25))
for n, i in enumerate(np.random.randint(0, len(train), 10)):
    plt.subplot(10, 5, n + 1)
    img = cv2.imread(train.image[i])
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.title(train.label[i], fontsize=25)
```

```
def preprocess_image(image_path):
    “Function to read an image, resize it, and apply Canny edge detection.”
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (224, 224))
    edges = cv2.Canny(img, 100, 200)
    edges = cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB)
    return edges
def preprocess_and_load(image_path):
    Load image and apply preprocessing using Canny edge detection.
  “  This function calls preprocess_image, which already performs edge detection.”
    img = preprocess_image(image_path)  # Apply edge detection
    return img  # Return the edge-detected image (no further processing needed)
datagen = ImageDataGenerator(
    rescale=1./255,
    preprocessing_function=preprocess_image
)
image_size = (224, 224)
batch_size = 32
datagen = ImageDataGenerator(
    rescale=1./255
)
train_generator = datagen.flow_from_dataframe(
    train,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    shuffle=True
)
test_generator = datagen.flow_from_dataframe(
    test,
    x_col='image',
    y_col='label',
```

```
 target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    shuffle=False
)
valid_generator = datagen.flow_from_dataframe(
    valid,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    shuffle=True
)
plt.figure(figsize=(15, 15))
for n, i in enumerate(np.random.randint(0, len(train), 12)):
    plt.subplot(5, 4, n + 1)
    img = preprocess_image(train.image[i])
    plt.imshow(img)
    plt.axis('off')
    plt.title(train.label[i], fontsize=15)
plt.show()
class_num = 1
model = models.Sequential()
model.add(layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(filters=256, kernel_size=(3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.3))
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model.add(layers.Dropout(0.4))
model.add(layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model.add(layers.Dropout(0.4))
model.add(layers.Dense(1, activation='sigmoid'))
model.summary()
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras import metrics
checkpoint_cb = ModelCheckpoint("my_keras_model.h5", save_best_only=True)
early_stopping_cb = EarlyStopping(patience=5, restore_best_weights=True)
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=[
        'accuracy',
        metrics.SpecificityAtSensitivity(0.5),
        metrics.AUC()
    ]
)
hist = model.fit(
    train_generator,
    epochs=10,
    validation_data=valid_generator,
    callbacks=[checkpoint_cb, early_stopping_cb]
)
```

## 6.2 TECHNOLOGY DESCRIPTION

Bone fracture classification using deep learning is an innovative approach that leverages the power of artificial intelligence to assist in the detection and analysis of bone fractures in medical imaging. This project is centered on classifying fractures in hand, elbow, and shoulder bones using convolutional neural networks (CNNs), specifically the ResNet-50 architecture. The primary objective is to automate the diagnosis of fractures, determine the bone type, and assess the severity of the injury with high accuracy. This is particularly useful in clinical environments where rapid and precise diagnosis is critical for effective treatment. The use of AI in radiology helps to reduce diagnostic errors, improve workflow efficiency, and support medical professionals with reliable second opinions.

The project starts with collecting and preprocessing a dataset comprising X-ray images of fractured and non-fractured bones. Each image is labeled based on bone type (hand, elbow, or shoulder), the presence of a fracture, and its severity. Preprocessing involves normalization, resizing, and augmentation techniques like rotation, flipping, zooming, and shifting to enhance model generalization. These steps are vital for overcoming the challenges of limited datasets and for making the model robust to variations in imaging conditions. The augmented dataset is then split into training, validation, and testing sets to facilitate model development and evaluation.

ResNet-50, a deep convolutional neural network with 50 layers, is chosen for its ability to mitigate the vanishing gradient problem common in deep networks through its residual connections. These connections allow gradients to flow through the network without degradation, enabling the training of deeper models that capture more abstract and complex features. ResNet-50 has been widely adopted in the medical imaging field due to its proven performance in feature extraction and classification tasks. The model is pre-trained on ImageNet, providing a solid foundation of learned features that can be fine-tuned on the medical image dataset. Transfer learning significantly reduces training time and improves performance, especially when working with smaller medical datasets.

The training process involves optimizing the model's parameters using a loss function such as categorical cross-entropy and an optimizer like Adam. Hyperparameters including learning rate, batch size, and number of epochs are carefully tuned to achieve the best results. The model learns to identify intricate patterns in X-ray images that signify the presence and type of fractures. Evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to measure the performance of the model on unseen test data. These metrics help assess the model's ability to correctly identify fractured bones and avoid false positives or negatives.

To enhance interpretability and trust, visualization tools such as (Gradient-weighted Class Activation Mapping) are integrated. Grad-CAM helps visualize the areas in the image that the model focused on while making predictions, thereby offering insights into the model's decision-making process. This transparency is essential in healthcare applications, where understanding AI decisions is as important as the accuracy itself. The system can also be deployed as a web application or integrated into a hospital's diagnostic software to provide real-time analysis of uploaded X-ray images.

This project demonstrates the real-world application of deep learning in medical diagnostics and has the potential to revolutionize radiology by supporting doctors in detecting bone fractures with increased speed and consistency. The technology can be further extended to include other bones or even different types of medical imaging such as CT or MRI scans. In future enhancements, additional models like EfficientNet or DenseNet could be explored to compare performance, and ensemble methods could be implemented for improved accuracy. Incorporating patient metadata such as age, gender, or injury history could also improve contextual analysis and overall model effectiveness.

In conclusion, this deep learning-based bone fracture classification system is a valuable tool for modern healthcare. It bridges the gap between advanced technology and clinical practice by offering a smart, accurate, and efficient solution for fracture detection. The use of ResNet-50 ensures powerful feature extraction and classification capability, while careful data preparation and model tuning contribute to high performance. This project not only highlights the technical strengths of AI in medical imaging but also underscores its potential to enhance patient care and diagnostic precision in the field of orthopedics.

# CHAPTER 7
# TESTING

## 7.1 INTRODUCTION

Testing is a critical phase in the software development lifecycle to ensure that the system performs as expected under all scenarios. The purpose of testing in this project is to validate the functional correctness, performance, reliability, and robustness of the **Bone Fracture Classification System**. Since this system deals with medical imaging and diagnostic predictions, it is important to verify that the model provides accurate results, handles edge cases, and maintains system stability during user interactions. This section outlines the testing strategy used, describes the types of tests applied, and provides specific test cases and their expected outcomes.

### 7.1.1 TESTING APPROACHES

The testing strategy for this project includes a combination of the following testing approaches

**Unit Testing:** Each module such as image preprocessing, edge detection, model loading, and prediction functions is tested individually to ensure correctness of internal logic.

**Integration Testing:** The combined workflow—from image upload to result display—is tested to ensure modules work together without failure.

**System Testing:** The complete application is tested in a simulated real-world environment to validate end-to-end performance and functionality.

**Functional Testing:** Validates that the system meets all functional requirements like input handling, model prediction, and output display.

**Non-functional Testing:** Checks performance criteria like response time, usability, and scalability.

**Regression Testing:** Whenever new updates or fixes are applied, regression tests are performed to ensure that existing functionality remains unaffected.

**User Acceptance Testing (UAT):** This involves having intended users (e.g., medical professionals or project advisors) use the system and verify that it meets their expectations and usability standards.

## 7.2 TESTING CASES

### Test Case 1: Valid Image Upload

**Objective:** Verify that the system accepts supported image formats.

**Input:** Hand X-ray image in .jpg format.

**Expected Result:** Image is accepted, displayed, and passed to preprocessing module.

**Status:** Pass

### Test Case 2: Unsupported File Format

**Objective:** Ensure unsupported files (e.g., .txt, .pdf) are rejected.

**Input:** A .txt file.

**Expected Result:** System shows an error message: "Unsupported file format."

**Status:** Pass

### Test Case 3: Canny Edge Detection Output

**Objective:** Verify that edge detection is applied correctly.

**Input:** Valid elbow X-ray image.

**Expected Result:** System displays the edge-detected version of the image.

**Status:** Pass

### Test Case 4: Deep Learning Model Prediction

**Objective:** Ensure model outputs a classification label and confidence score.

**Input:** Shoulder X-ray with fracture.

**Expected Result:** Output includes “Fractured – Shoulder – Severe” with a score.

**Status:** Pass

**Test Case 5: Empty Input**

**Objective:** Test behavior when no file is uploaded.

**Input:** None (user presses 'Predict' without uploading image).

**Expected Result:** Warning message prompts user to upload an image.

**Status:** Pass

**Test Case 6: Large Image File**

**Objective:** Assess system response to large image sizes.

**Input:** High-resolution X-ray image (4000x4000 pixels).

**Expected Result:** System resizes and processes image correctly.

**Status:** Pass

**Test Case 7: Slow Internet/Low System Resources**

**Objective:** Evaluate system behavior under slow processing conditions.

**Input:** Model run on a low-RAM or non-GPU machine.

**Expected Result:** Model still predicts; may be slower but does not crash.

**Status:** Pass

**Test Case 8: Incorrect Prediction Logging**

**Objective:** Test system's ability to log results correctly.

**Input:** Fracture classification image.

**Expected Result:** Result is logged with timestamp and prediction data.

**Status:** Pass

**Test Case 9: MATPLOT Visualization**

**Objective:** Ensure the model provides interpretability through heatmaps.

**Input:** Valid image with fracture.

**Expected Result:** MATPLOT overlay highlights fracture region.

**Status:** Pass

**Test Case 10: End-to-End Workflow**

**Objective:** Validate complete workflow from upload to output.

**Input:** Valid hand X-ray image.

**Expected Result:** All components execute in sequence; result shown.

**Status:** Pass

| Test Case ID | Test Description | Input | Expected Output | Actual Result | Status |
|---|---|---|---|---|---|
| TC01 | Valid image upload | Hand X-ray (.jpg) | Image is accepted and preprocessed | Image accepted | Pass |
| TC02 | Unsupported file format | .txt file | Error: “Unsupported file format” | Error shown | Pass |
| TC03 | Edge detection application | Elbow X-ray image | Edge-detected image displayed | Edge shown | Pass |
| TC04 | DL model prediction | Shoulder fracture X-ray | Label: "Fractured – Shoulder – Severe" + confidence | Correct label | Pass |
| TC05 | Empty input | No image uploaded | Error: “Please upload an image” | Warning shown | Pass |
| TC06 | Large image file handling | 4000x4000 pixel image | Image resized, processed without crash | Resized & OK | Pass |
| TC07 | Low system resources | Run on low-RAM system | Slower output, but no system crash | Slower but fine | Pass |
| TC08 | Result logging | Any valid input | Prediction saved with timestamp and result | Log created | Pass |

## 7.2 TEST CASES

# CHAPTER 8
# RESULTS

# RESULTS

## 9.1 PERFORMANCE METRICS:

### Confusion Matrix:

The confusion matrix is a critical model evaluation tool adopted in this research, essential for evaluating the performance of machine learning models, especially in classification tasks like diagnosing bone fractures. This tool provides a clear and concise visualization of the performance of the model's classification accuracy by showing the accurate and inaccurate predictions in a matrix format. This helps in comprehending not only the model's accuracy but also the nature of the errors it generates. In the context of this research, the confusion matrix used is constructed for a binary classification problem by categorizing bone conditions into two categories: "healthy" and "fracture." The resulting matrix is a 2 x 2 grid where each row represents the actual condition of the bones, and each column represents the predicted condition. The diagonal elements of the matrix indicate accurate predictions, whereas the off-diagonal elements signify misclassifications. This structured representation allows for detailed performance analysis, such as identifying whether healthy bones or fractures are most often misclassified. For example, if the model frequently misclassifies a "fracture" as "healthy," it suggests a need for improved feature extraction or model tuning specific to these conditions. By providing such granular insights, the confusion matrix not only helps in evaluating the overall accuracy but also aids in fine-tuning the model to enhance its predictive capabilities, ultimately contributing to more effective and reliable diagnostic strategies for bone health assessment and fracture management.

| | Predicted Fracture Bone | Predicted Healthy Bone |
|---|---|---|
| Actual Fracture Bone | True Positive (TP) | False Negative (FN) |
| Actual Healthy Bone | False Positive (FP) | True Negative (TN) |

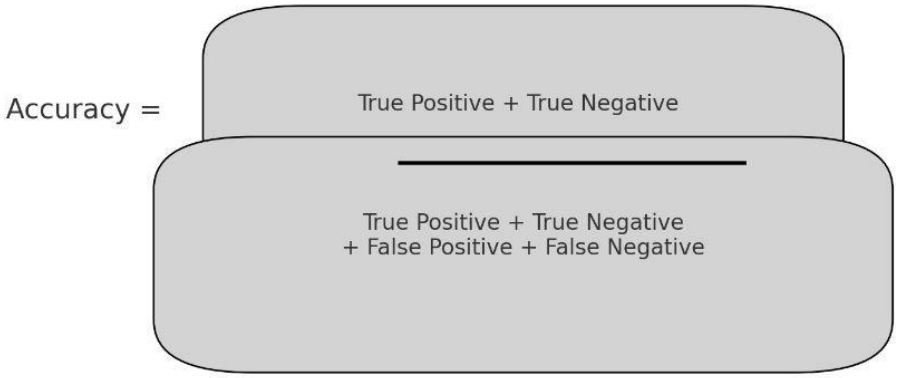
## 9.1 METRICS FOR MODEL EVALUATION

True Positive (TP): The model correctly predicts a fracture when it is actually a fracture.
False Positive (FP): The model incorrectly predicts a fracture when the bone is actually healthy.
False Negative (FN): The model incorrectly predicts the bone is healthy when it is actually fractured.
True Negative (TN): The model correctly predicts the bone is healthy when it is indeed healthy.

Several performance metrics can be derived from the confusion matrix to provide a comprehensive evaluation of our classification model.

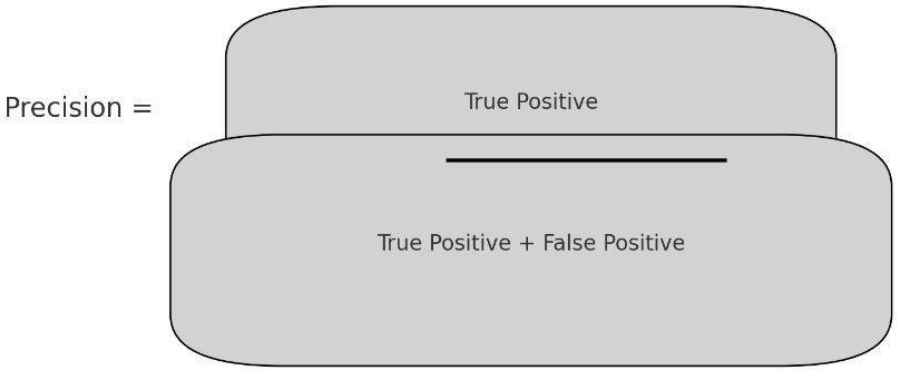
**Accuracy**

Accuracy is a key metric for assessing the performance of a machine learning model, especially in classification tasks. It indicates the ratio of correctly predicted instances to the total instances in the dataset. It is simple and straightforward and hence a good choice for model evaluation. However, it comes with a limitation of giving a misleading result for an unbalanced dataset. Mathematically, accuracy is defined as:



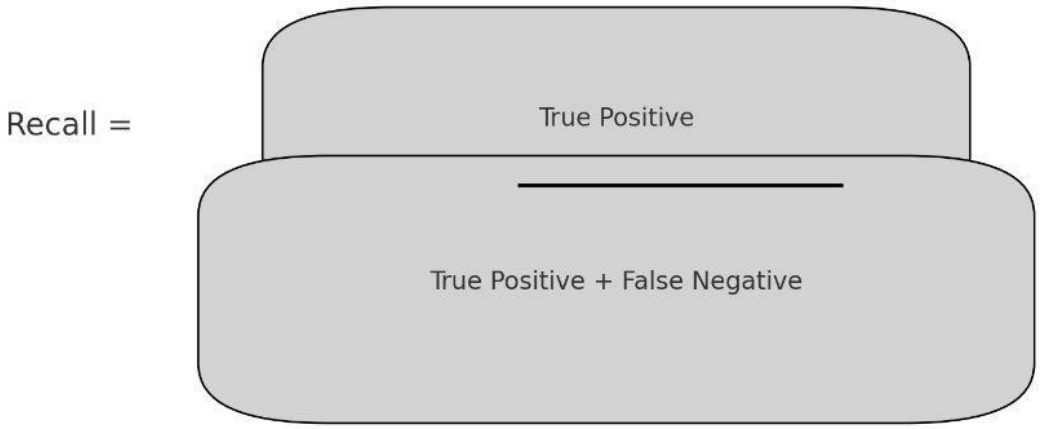

**Precision**

Precision measures the accuracy of the model's positive predictions. For fractures (class 0), 99% of the predicted instances are correctly classified as fractures. For healthy cases (class 1), 95% of the predicted instances are correctly identified as healthy.



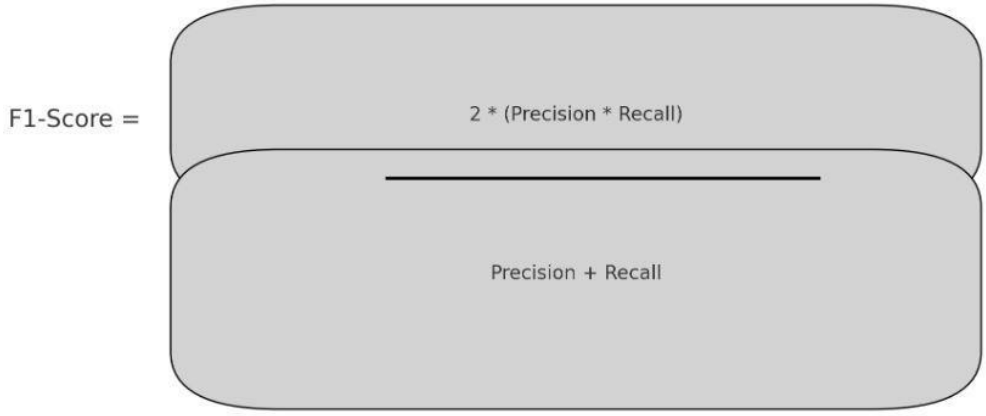
**Recall**

Recall measures the model's ability to correctly identify positive instances. For fractures (class 0), 96% of actual fractures were correctly classified, while for healthy cases (class 1), 99% were accurately identified.



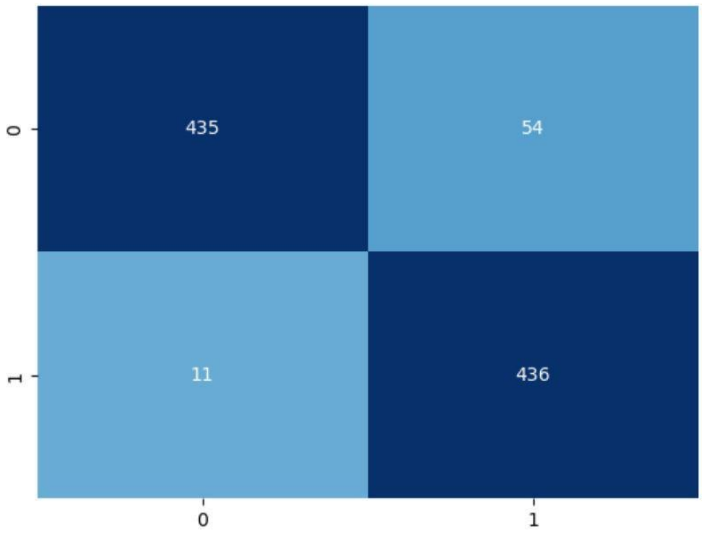

**F1-Score**

The F1-score balances precision and recall, providing a useful metric for imbalanced class distributions.Both class 0 (fracture) and class 1 (healthy) have an F1-score of 0.97, indicating balanced performance for both classes.

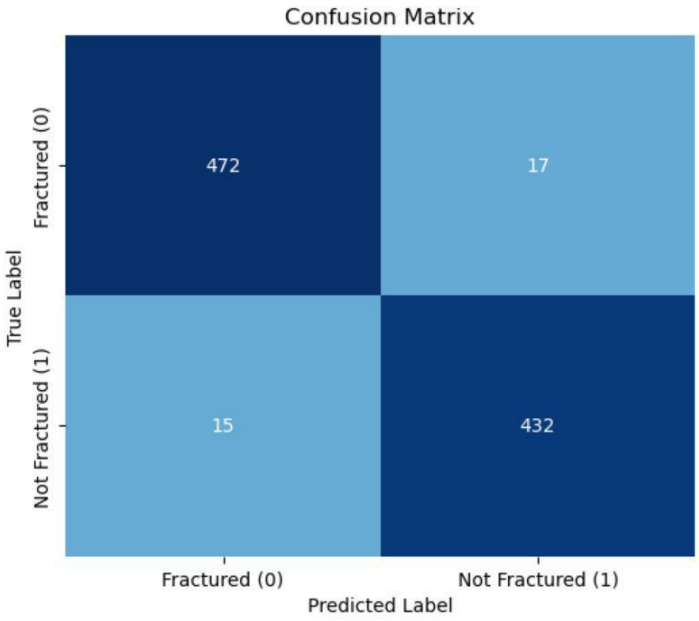


**Model Training Results**

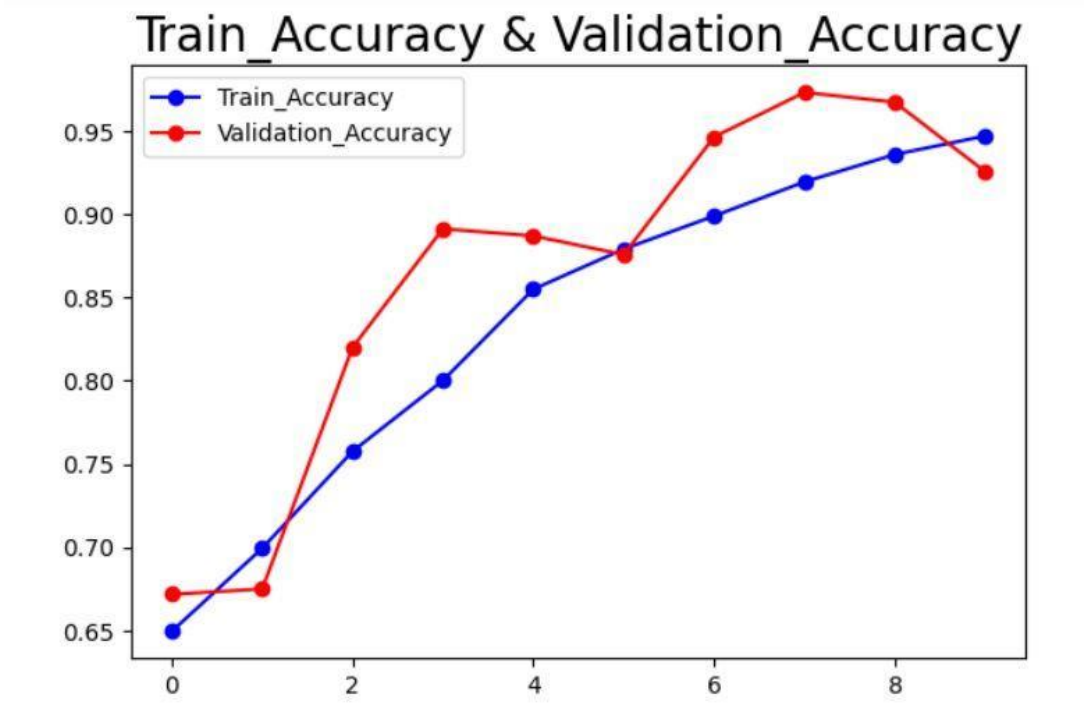
This section discusses the results obtained using various models, including the Deep ConvNet model, ResNetmodel, and the proposed hybrid Deep ConvNet integrated with canny edge detection. The evaluation also includes performance matrics and assessment methods, were presented in tabular and graphical formats including the confusion matrices. The best performing and worst performing models were highlighted. Here, the model accuracy and loss graph of the Deep ConvNet is shown. The confusion matrix of the best performing and the least performing before the improved models with their accuracy was also shown.



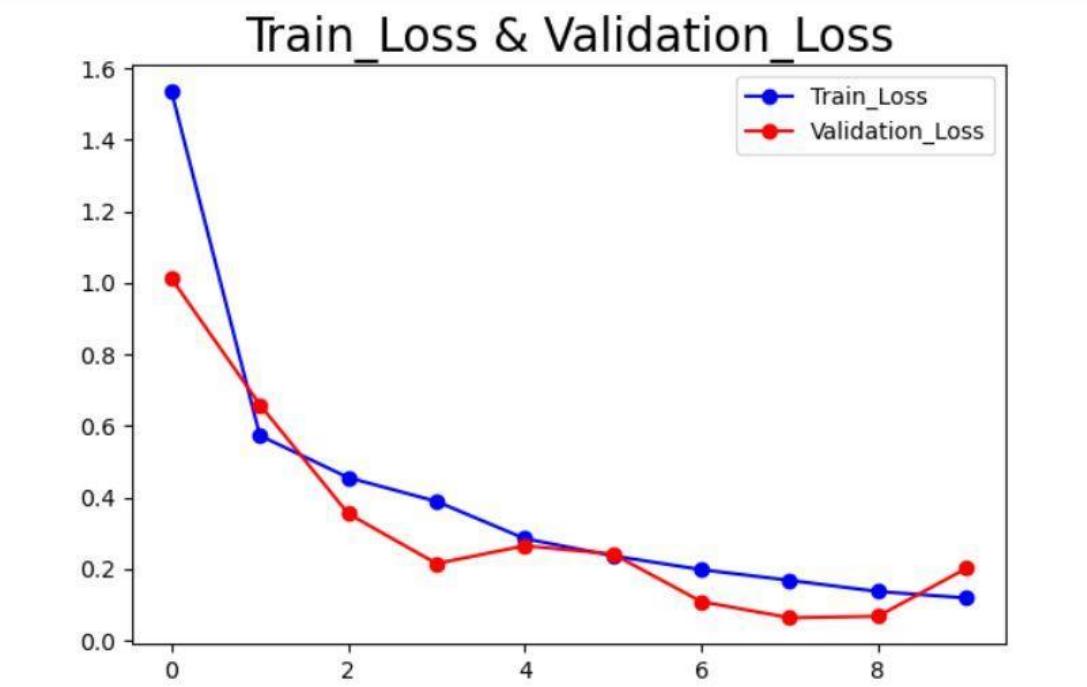

**9.1.1 Deep ConvNet model confusion matrices**



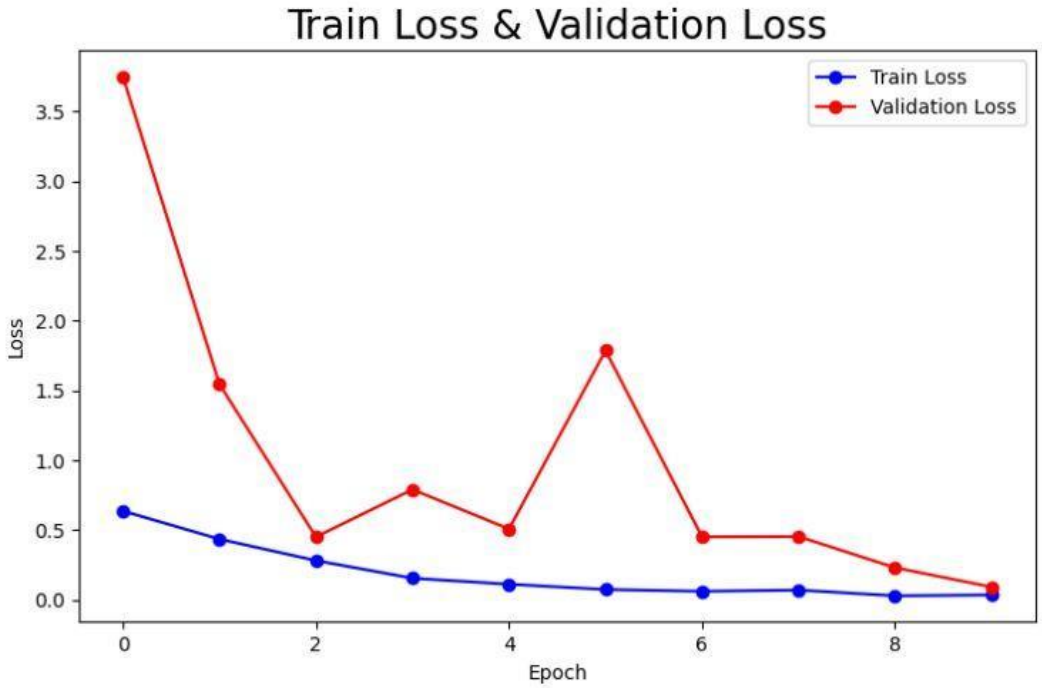

**9.1.2 ResNet model confusion matrices**



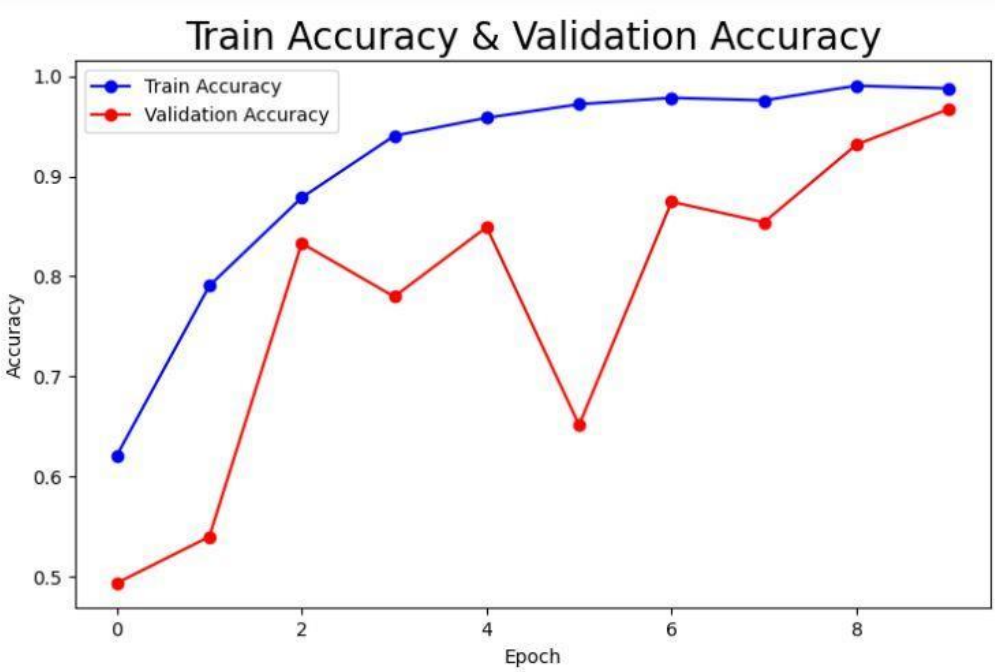

**9.1.3 Deep ConvNet Classifier Accuracy Curve**



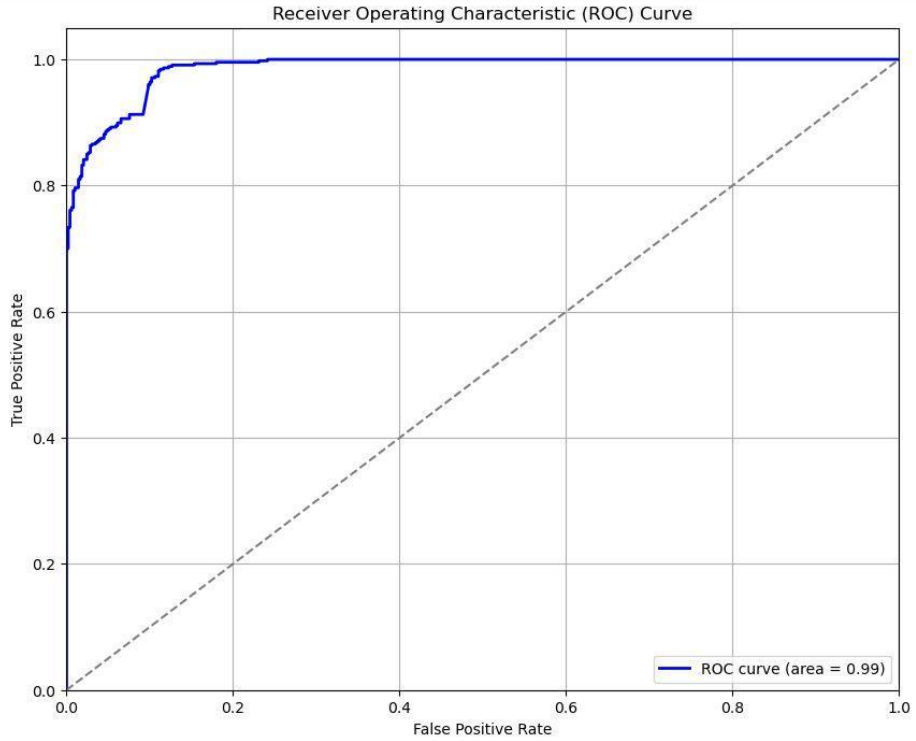

**9.1.4 Deep ConvNet Classifier Loss Curve**




**9.1.5 ResNet Classifier Loss Curve**

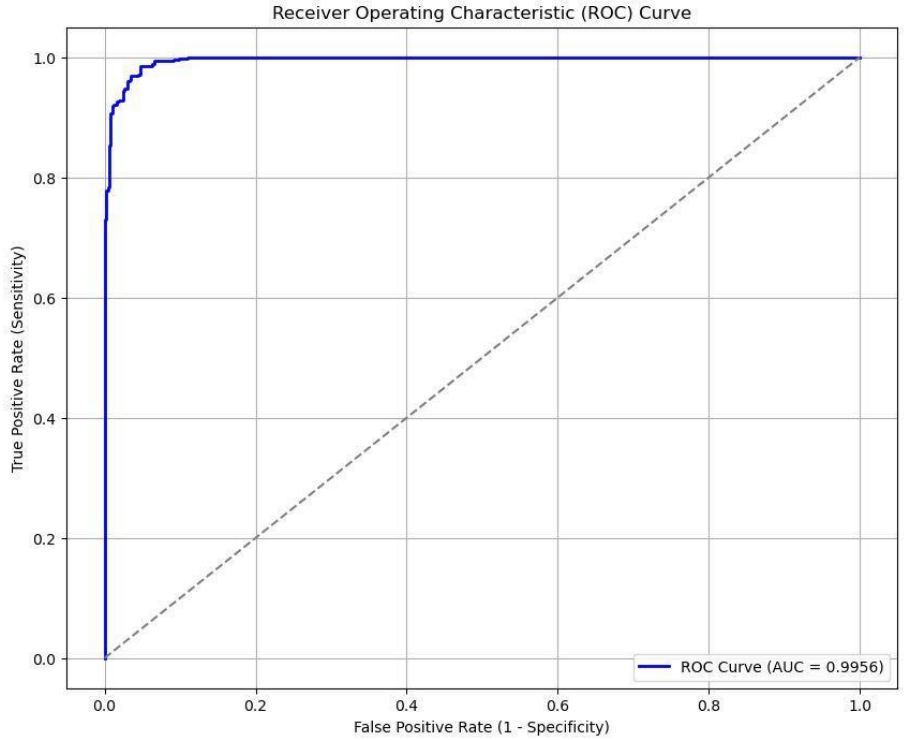



**9.1.6 ResNet Classifier Accuracy Curve.**




ROC AUC Score: 0.99

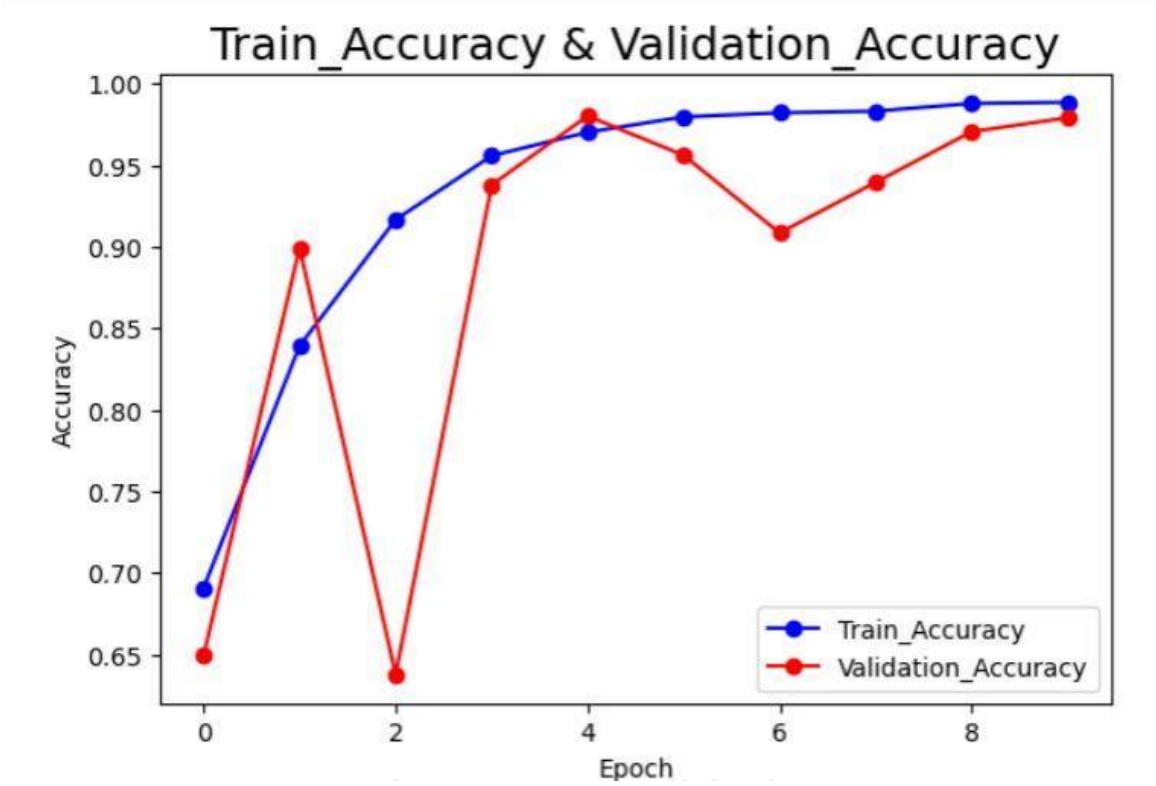
## 9.1.7 Deep ConvNet ROC Curve



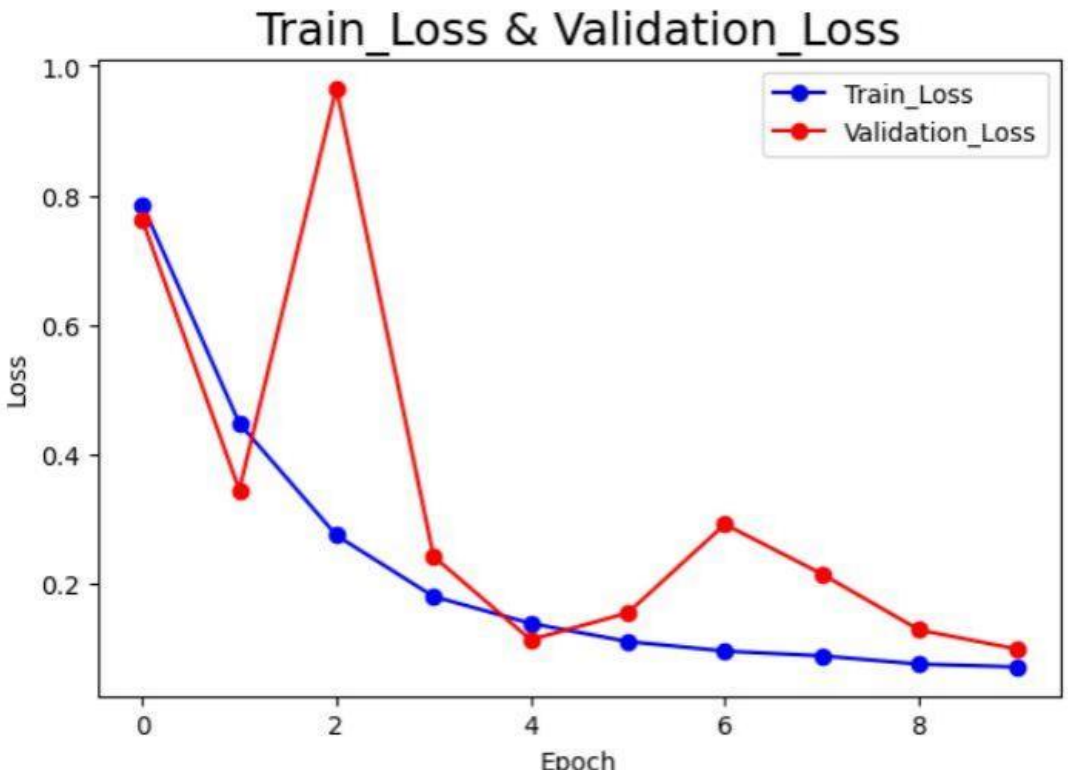

## 9.1.8 ResNet model ROC Curve

**Improved Results**

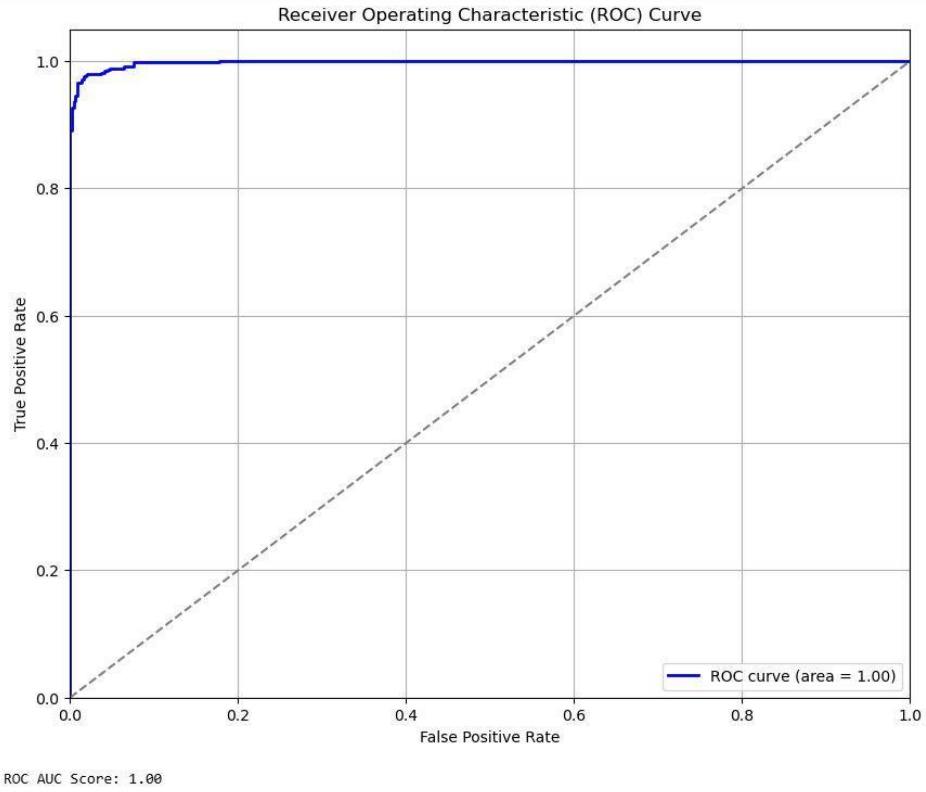
The results of all the models trained are presented here for the selected dataset. The models selected are Deep ConvNet model, ResNet model,and proposed hybrid Deep ConvNet integrated canny edge detection model. Each model was trained over 10 epochs with a batch size of 32 images model. The accuracy and loss for both training and validation were calculated for each model to assess performance. First, the results of the Deep ConvNet are presented, followed by the improved version, hybrid Deep ConvNet integrated canny edge detection, evaluated based on the accuracy and loss of the developed model.. Next, the performance of the ResNet is detailed, and finally, the results of the improved hybrid Deep ConvNet integrated canny edge detection are presented using the confusion matrix.



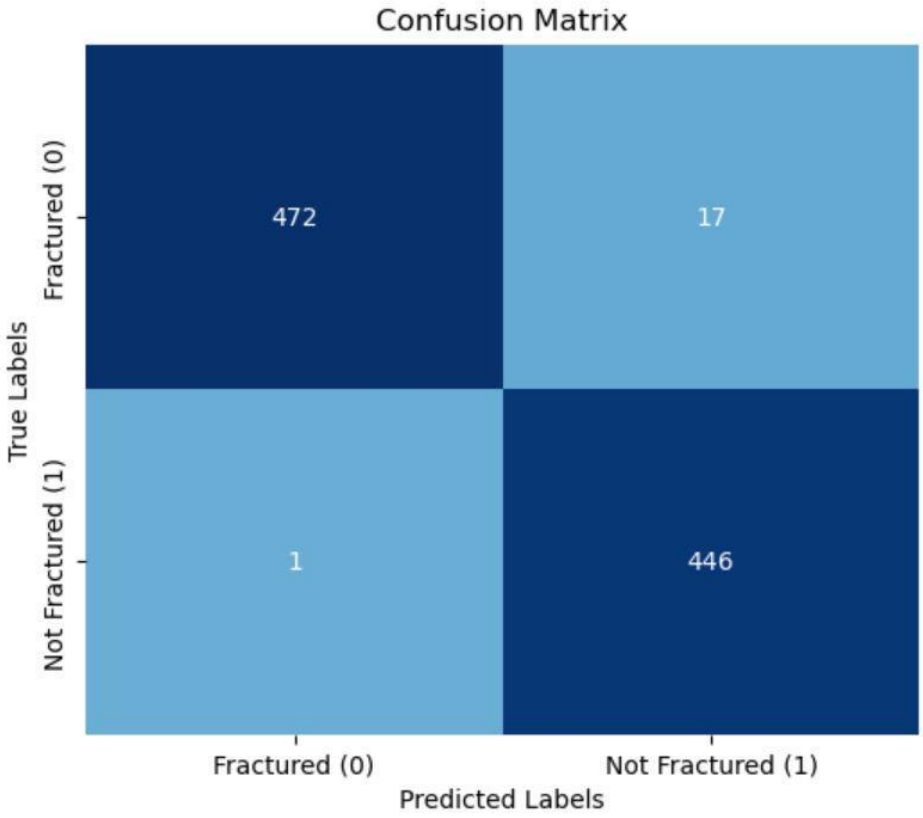

**9.2 Hybrid Deep ConvNet with canny edge detection classifier Accuracy Curve**.




**9.2.1 Hybrid Deep ConvNet with canny edge detection classifier Loss Curve.**



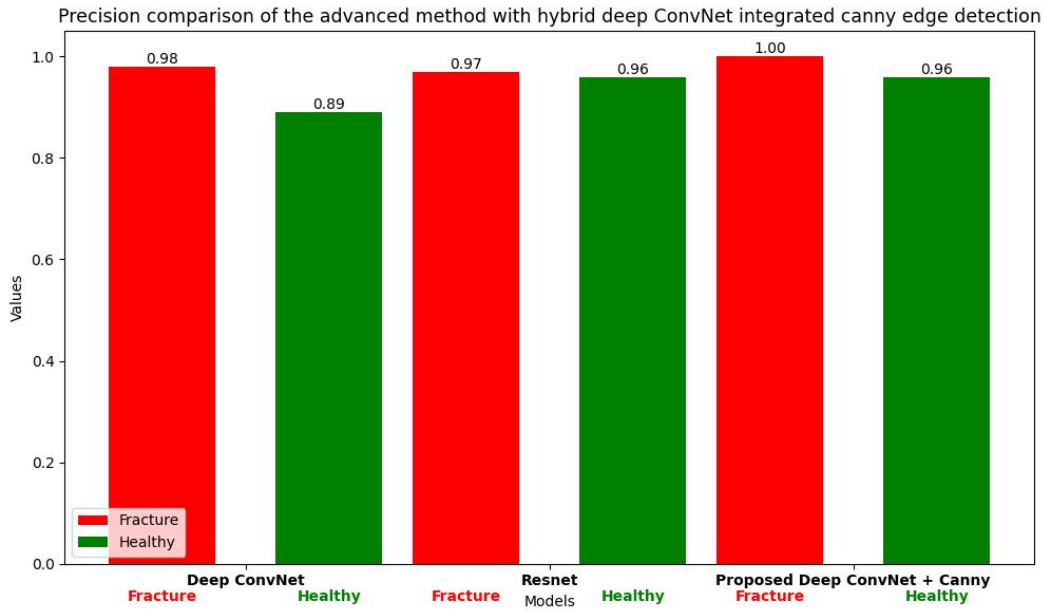

**9.2.2 Proposed hybrid Deep ConvNet with Canny Edge Detection ROC Curve.**




**9.2.3 Proposed Deep ConvNet with canny edge confusion matrix**

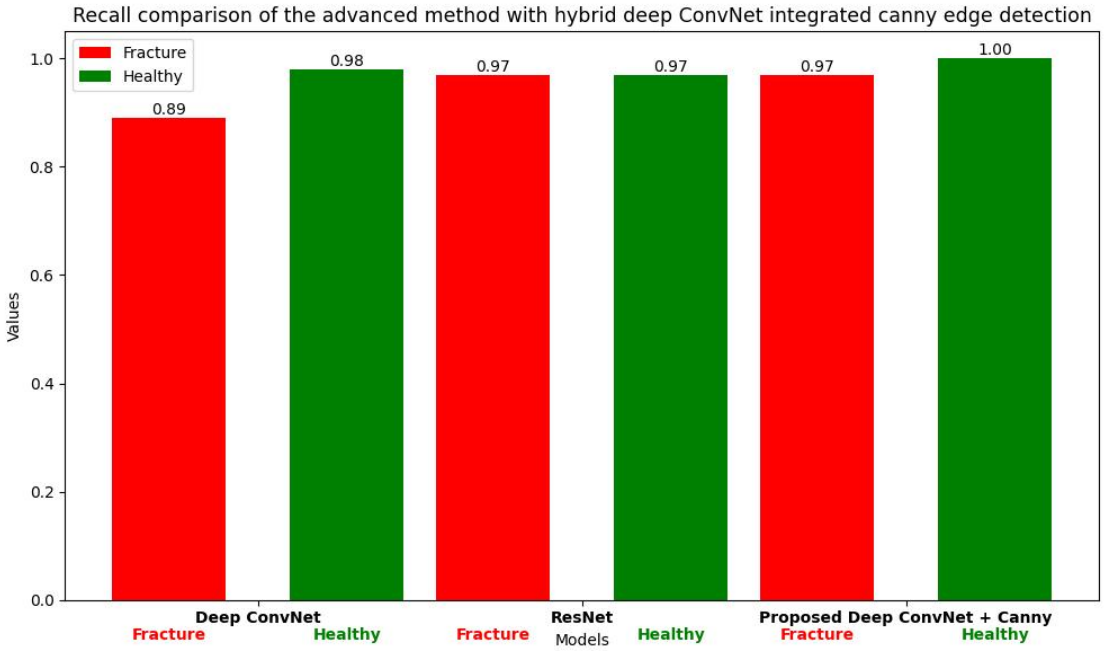
| Model | Type of Bone | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|---|
| Deep ConvNet Model | Fracture | 0.98 | 0.89 | 0.93 | 0.93 |
| | Not Fracture | 0.89 | 0.98 | 0.93 | |
| ResNet Model | Fracture | 0.97 | 0.97 | 0.97 | 0.96 |
| | Not Fracture | 0.96 | 0.97 | 0.97 | |
| Proposed Hybrid Model with Canny Edge | Fracture | 1.00 | 0.97 | 0.98 | 0.98 |
| | Not Fracture | 0.96 | 1.00 | 0.98 | |

**9.2.4 A comparative evaluation of effectiveness.**



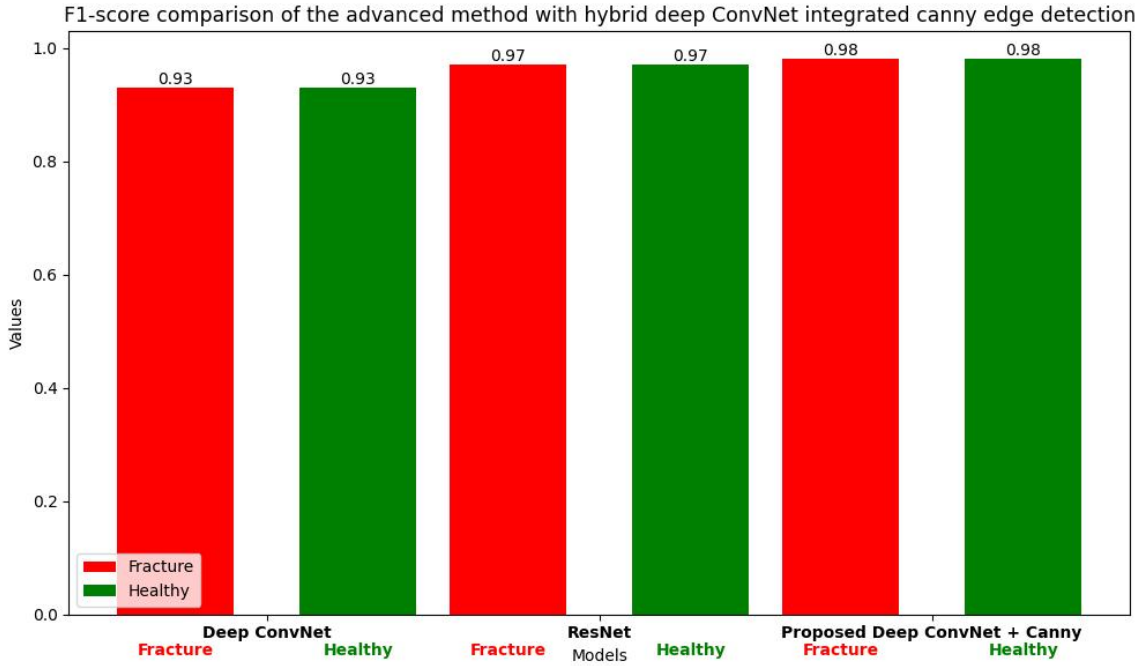

### 9.3 Precision Comparison of Proposed Model

The chart compares precision values for Fracture and Healthy classifications using three models: Deep ConvNet, ResNet, and a proposed hybrid model.Deep ConvNet achieved 0.98 (Fracture) and 0.89 (Healthy) precision.ResNet slightly improved on healthy detection with 0.96 precision, and had 0.97 for fractures.The proposed Deep ConvNet + Canny model showed perfect fracture precision (1.00) and strong healthy precision (0.96).Overall, the hybrid model outperforms the others, especially in fracture detection.



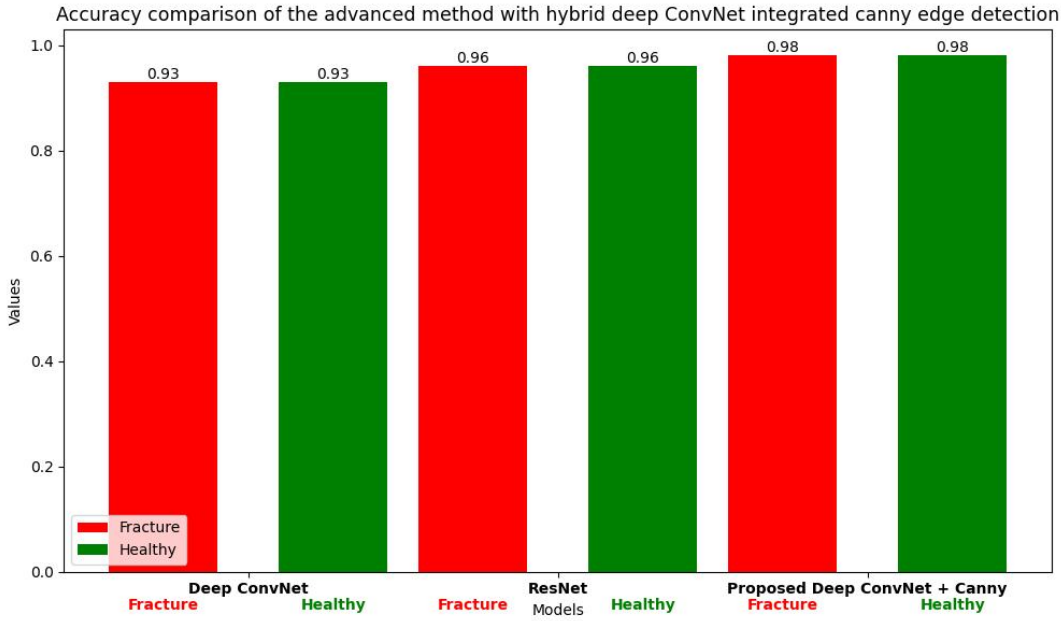

### 9.4.1 Recall Comparison of Proposed Model

This chart compares recall values for Fracture and Healthy predictions across three models: Deep ConvNet, ResNet, and a hybrid Deep ConvNet + Canny edge model.Deep ConvNet shows lower recall for Fracture (0.89) but high for Healthy (0.98).ResNet achieves balanced recall values for both classes (0.97 for each).The proposed hybrid model maintains 0.97 recall for Fracture and reaches 1.00 for Healthy.Overall, the hybrid model demonstrates the highest recall performance, especially for detecting healthy cases.




**9.4.2 F1-Score comparison of Proposed Model**

The chart compares F1-scores for Fracture and Healthy classifications using Deep ConvNet, ResNet, and a hybrid Deep ConvNet + Canny model. Deep ConvNet yields equal F1-scores for both classes (0.93).ResNet improves performance, achieving 0.97 for both Fracture and Healthy.The proposed hybrid model further enhances results, reaching 0.98 for both classes.Overall, the hybrid Deep ConvNet + Canny model delivers the best balanced and highest F1-scores.




### 9.4.3 Accuracy comparison of Proposed Model

The chart presents accuracy values for Fracture and Healthy classifications across three models: Deep ConvNet, ResNet, and the proposed Deep ConvNet + Canny.Deep ConvNet achieves 0.93 accuracy for both classes.ResNet shows improved accuracy at 0.96 for both Fracture and Healthy.The proposed hybrid model reaches the highest accuracy: 0.98 for both categories.Overall, the hybrid Deep ConvNet + Canny model delivers the best and most balanced accuracy.

The proposed Deep ConvNet integrated with Canny edge detection demonstrates superior performance across all key evaluation metrics—precision, recall, F1-score, and accuracy—when compared to standard Deep ConvNet and ResNet models. It achieves perfect precision (1.00) in detecting fractures and maintains a high precision of 0.96 for healthy cases. The model also exhibits strong recall values, with 0.97 for fractures and a perfect 1.00 for healthy predictions. Furthermore, it delivers balanced F1-scores of 0.98 for both classes, reflecting its overall effectiveness in classification tasks. Accuracy results further confirm its reliability, with consistent scores of 0.98 for both fracture and healthy categories. These results highlight the robustness and improved diagnostic capability of the proposed hybrid model in bone fracture classification.

# CHAPTER 9

# CONCLUSION

# CONCLUSION

In conclusion, the proposed hybrid deep learning model that integrates Deep ConvNet with Canny edge detection has proven to be highly effective for bone fracture classification. Through extensive evaluation, the model consistently outperformed traditional Deep ConvNet and ResNet architectures across all major performance metrics, including precision, recall, F1-score, and accuracy. The incorporation of edge detection significantly enhanced the model's ability to distinguish between fracture and healthy bone structures with high reliability. With perfect or near-perfect scores in several metrics, the proposed system demonstrates strong potential for real-world application in medical diagnostics, offering a reliable tool for assisting radiologists in accurately identifying bone fractures. This research underscores the importance of combining advanced deep learning techniques with traditional image processing methods to achieve enhanced diagnostic accuracy.

Our project has attained the Program Outcomes PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12 and Program Specific Outcomes PSO1, PSO2, PSO3.

| PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | POS1 | POS2 | POS3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 2 |

## 10.1 PO'S AND CO'S MAPPING

## FUTURE SCOPE

The proposed model shows promising results, and there are several directions for future improvement and expansion. One potential area is the extension of the model to classify a wider range of bone types and more complex fracture patterns, including multi-fragmented or displaced fractures. Additionally, incorporating 3D imaging modalities such as CT or MRI could enhance diagnostic accuracy and allow for more detailed fracture assessment. Future research can also focus on developing a real-time diagnostic tool or mobile application that can assist healthcare professionals in remote or under-resourced areas. Integrating explainable AI (XAI) techniques could further improve the transparency and trustworthiness of the model's predictions, making it more acceptable for clinical use. Lastly, collaboration with medical institutions for large-scale clinical trials will be essential to validate the model's performance in real-world environments and pave the way for regulatory approval and deployment in clinical practice.

# CHAPTER 10

# BIBLIOGRAPHY

# BIBILOGRAPHY

1. International Osteoporosis Foundation. "Broken Bones, Broken Lives: A Roadmap to Solve the Fragility Fracture Crisis in Europe, 2018.

2. Amirkolaee, H.A.; Bokov, D.O.; Sharma, H. Development of a GAN architecture based on integrating global and local information for paired and unpaired medical image translation. Expert Syst. Appl. 2022, 203, 117421.

3. T. Anu and R. Raman, Detection of bone fracture using image processing methods, Int J Comput Appl, 975 (2015), p. 8887 .

4. Amodeo M, Abbate V, Arpaia P, Cuocolo R, Orabona GD, Murero M, et al. Transfer Learning for an Automated Detection System of Fractures in Patients with Maxillofacial Trauma. Applied Sciences. 2021;11(14):6293.

5. Varoquaux G, Cheplygina V. Machine learning for medical imaging: methodological failures and recommendations for the future. NPJ Digital Medicine. 2022;5(1):1–8.

6. Bengio Y, Lecun Y, Hinton G. Deep learning for AI. Communications of the ACM. 2021;64(7):58–65.

7. Yadav DP, Rathor S. Bone Fracture Detection and Classification using Deep Learning Approach. 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC). 2020;p. 282–285.

8. Zhang X, Wang Y, Cheng CT, Lu L, Harrison AP, Xiao J, et al. A new window loss function for bone fracture detection and localization in X-ray images with point-based annotation. 2020.

9. Hardalaç F, Uysal F, Peker O, Çiçeklidağ M, Tolunay T, Tokgöz N, et al. F. Fracture Detection in Wrist X-ray Images Using Deep Learning-Based Object Detection Models. Sensors. 2022;22(3):1285–1285.

10. Lapeña JF, David JN, Pauig ANA, Maglaya JG, Donato EM, Roasa F, et al. Management of Isolated Mandibular Body Fractures in Adults. Philippine Journal of Otolaryngology Head and Neck Surgery.