Cairo University

Faculty of Computers and Artificial Intelligence

FRACSCAN

DEEP LEARNING FOR BONE FRACTURE DETCTION

Under Supervision of:

DR: Nour Eldeen Mahmoud

Eng. Belal Hassan

Prepared By:

Shreen Osama Abdelrahman	20200263
Mahmoud Ayman Abdelhamid	20200494
Ahmed Yasser Ahmed Meshref	20200061
Abdelrahman Ahmed Mihoub	20200280
Ahmed Ashraf Zaki Omar	20200807
Youssef Mohamed Ahmed	20200661

Final Documentation

Artificial Intelligence Department

Academic Year 2023/2024

Table of Contents

List of Figures

List of Tables

Abstract

Our project discusses sophisticated issue which healthcare specifically in Orthopedics field, big segments of people suffering from bones fractures and can be aggravated leading to big injuries, Vertebral or spinal fractures are the most common fractures occurring in 30-50% of people over the age of 50 and result in significantly increased morbidity and mortality. Hip fractures, while occurring less frequently, are the most devastating fractures as 20% of those who suffer from a hip fraction die within 6 months, an increase of 113% compared to 1990 and Predictions in the European Union indicate that the total number of fractures in men will increase by 34% by 2025 to reach 1.6 One million cases per year[1]and In the US there are about 2 million fractures each year.

However, we're concentrating on dealing with this issue specifically bones fractures helping physicians and radiologists to examine, detect and cure these diseases before aggravation, Using AI with computer vision and advanced object detection to classify, localize and detect the fractures helping physicians to take corrective action for patients.

Patients usually do X-Rays, CT-Scan and MRI but we have focused on X-Rays images trying to detect fractures and its severities handling these constitute sophisticated challenges also finding dataset full of kinds of fractures for lower and upper extremities from AP, lateral and oblique bone views.

Most of research in this case in scientific say however, no real products have been published in the market so, we have to be more responsible for being one of the first publishers and producers in this market make it simpler for physician, radio centers and also patients and using our AI medical assistant to make earlier checkup and prediction to save our people and curing them rapidly.

Chapter 1: Introduction

1.1 Background:

The field of diagnostic radiology is an integral part of modern health care, as X-ray imaging is an essential means of detecting skeletal deformities, especially fractures.

The evolution of diagnostic radiology traces back through a fascinating history, illuminating its pivotal role in modern healthcare. Initially emerging with the discovery of X-rays by Wilhelm Roentgen in 1895, diagnostic radiology has since undergone transformative advancements. The pioneering days of X-ray imaging paved the way for revolutionary breakthroughs, shaping the landscape of medical diagnostics.

- Brief History of Diagnostic Radiology:

1. *Pioneering Discovery (1895)*: Wilhelm Roentgen's accidental discovery of X-rays marked the birth of diagnostic radiology, opening new possibilities in medical imaging.

2. *Rapid Advancements (20th Century)*: The 20th century witnessed rapid technological advancements, introducing innovations such as computed tomography (CT), magnetic resonance imaging (MRI), and digital radiography, expanding the diagnostic capabilities of the field.

3. *Digital Era (21st Century)*: The 21st century ushered in the era of digital radiology, transforming traditional film-based methods into sophisticated digital platforms, enhancing image quality and accessibility.

The importance of diagnostic radiology in contemporary healthcare cannot be overstated. It serves as a linchpin for detecting a myriad of medical conditions, providing clinicians with non-invasive insights into the inner workings of the human body. Radiology centers, entrusted with the task of analyzing a vast influx of X-ray images daily, play a critical role in facilitating accurate and timely patient diagnoses.

- Importance of Diagnostic Radiology:

1. *Non-Invasive Insight*: Diagnostic radiology offers a non-invasive means to visualize internal structures, enabling the detection of a wide array of medical conditions without the need for surgery.

2. *Early Disease Detection*: The ability to detect diseases and abnormalities in their early stages enhances the chances of successful treatment and improved patient outcomes.

3. *Guidance for Treatment Planning*: Radiological images serve as crucial guides for clinicians, aiding in treatment planning and surgical interventions.

1.2 Problem Statement:

However, the traditional manual approach to radiological image interpretation poses challenges, especially in the face of the rapid pace and complexity of modern healthcare. This is where our project comes into play, as it takes advantage of artificial intelligence to enhance diagnostic radiology capabilities, specifically in the field of fracture diagnosis.

- Challenges in the traditional approach:

1. *Time constraints*: Manual interpretation is time-consuming, which hampers the rapid analysis required to diagnose a patient in a timely manner.

2. *Human error*: Subjectivity leads to the risk of censorship and misinterpretation because the doctor, even if he is very experienced, is exposed to factors such as loss of concentration, etc., which confirms the need for an objective and accurate approach that greatly helps doctors in diagnosis.

In addition, medical students in the Department of Radiology in particular face a great challenge in learning because of the necessity of having an accurate diagnosis for each case studied, and in many cases, this is not widely available. Therefore, it was useful to have a solution that could diagnose any fracture within seconds and give accurate results that students could rely on while studying.

1.3 Scope:

Our project is directed primarily to the medical field to assist in radiological diagnosis in general, whether medical students or doctors during work, and secondarily, so that anyone can obtain an initial diagnosis until they go to a specialist to review it and take remedial steps later.

1.4 Objectives:

Our project aims to seamlessly integrate cutting-edge technology with the rich history and importance of diagnostic radiology. By automating fracture diagnosis, it not only addresses the immediate clinical needs but also contributes to the ongoing legacy of diagnostic radiology, providing a solution that aligns with its historical significance and contemporary importance.

- Student-Centric Challenges:

1. *Limited Case Availability*: The project recognizes the challenges faced by medical students, offering a solution that enhances their learning experience by providing instant and accurate fracture diagnoses.

2. *Need for Rapid Learning*: The dynamic nature of medical education demands a solution that expedites the learning process, offering a diverse database of annotated fractures for comprehensive study.

1.5 Significance of the Study:

This project, harmonizing with the historical trajectory and vital role of diagnostic radiology shapes the educational landscape for future medical professionals; by combining rapid diagnosis for professionals and an extensive learning resource for students, our solution aims to elevate the efficiency and effectiveness of fracture diagnostics in both practice and academia.

1.6 Methodology definition:

Methodology Definition:

For this system, we will follow a linear model consisting of 8 sequential stages, each meticulously designed to ensure the efficacy and reliability of the overall process:

1- Problem Analysis:

Firstly, we begin with the problem analysis phase, which includes understanding and defining the problem. This involves a comprehensive examination of the overarching problem statement.

2- Data Collection and Preprocessing:

Secondly, the data collection and processing stage, in which we explain the datasets available to train and test the model and how they were processed to become suitable for the model.

3-Model Design:

The third stage is the model design phase, where we design the model structure and high-level technical details of the project such as the class diagram. In addition, we plan the technologies we will use such as programming languages, database,

4- Model Implementation:

In the fourth stage, model implementation, we code and implement the model, discussing the models and algorithms we used. This involves translating the designed model into executable code or a tangible system, emphasizing efficiency, scalability, and maintainability.

5-Model Training:

In the fifth stage, model training, we train the model on the data that we collected and previously processed. Training parameters are fine-tuned, and iterative adjustments are made to optimize model performance and generalization capabilities.

6- Model Testing:

The sixth stage is the testing phase, where we thoroughly assess the model after the training process. During this stage, we focus on evaluating the model's performance and accuracy. We aim to improve the results by experimenting with various variables, such as adjusting hyperparameters, modifying input data, and employing different evaluation metrics. By iterating through these experiments, we strive to enhance the model's overall performance and robustness. Our goal is to achieve the best possible outcome before moving forward to the deployment stage, ensuring that the model is both reliable and effective for its intended application

7-Deployment: The deployment phase is when the system is released into the relevant environment for use by the public. This stage involves careful integration with existing systems and infrastructure to ensure seamless functionality and compatibility.

8-Maintenance:

Finally, the maintenance phase is when the system is in use, and change is needed. The lifecycle of the deployed model is continuously monitored, managed, and optimized to sustain its performance and relevance over time, with regular updates, retraining, and adaptation to evolving data dynamics or business requirements undertaken as necessary.

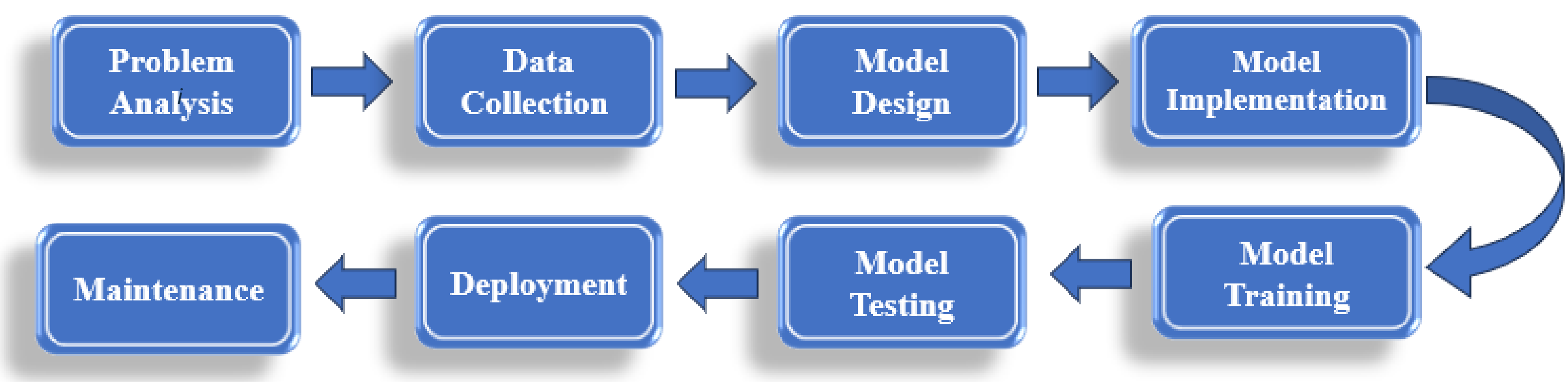


figure 1 methodology life cycle

Chapter2: Related work

1)IBFDS: Intelligent bone fracture detection system:

In this paper, an intelligent classification system for bone fractures is developed. The proposed system consists of two main phases, which are the processing phase and the classification phase. In the image processing phase, the images are processed using techniques such as Haar Wavelet transforms, as well as SIFT as feature extractor. These techniques are utilized to enhance the quality of the images and to extract the fractured part of the bone. At the end of this phase, the images are ready to be fed into the new phase, which is the neural network. MATLAB programming language is used for implementing and simulating the suggested system (MATLAB 2013a). As stated earlier, SIFT uses a method to detect interest points from a grey-level image that uses image intensities which are accumulated in the image structures. The purpose of this research is to evaluate the effectiveness of a backpropagation neural network in recognizing different bone fractures. As stated in the previous section, the developed framework consists of image and classification phases. In the processing phase, the fracture images are preprocessed. Then the feature extraction using SIFT takes place.

Intelligent Bone Fracture Detection System (IBFDS) uses two conventional 3-layer back propagation neural networks (BPNN) with 1024 input neurons. Output neurons classify the bone using binary coding: [1 0] for the bone with fracture, and [0 1] for the bone without fracture. The sigmoid activation function is used for activating neurons in the hidden and the output layers, Due to the implementation simplicity, and the availability of sufficient “input – target” database for training, the use of a back propagation neural network which is a supervised learner, has been preferred.

Training and testing (generalization) are comprised for this phase. The available bone image database is organized as follows:

1. Training image set: 30 images.

2. Testing image set: 70 images.

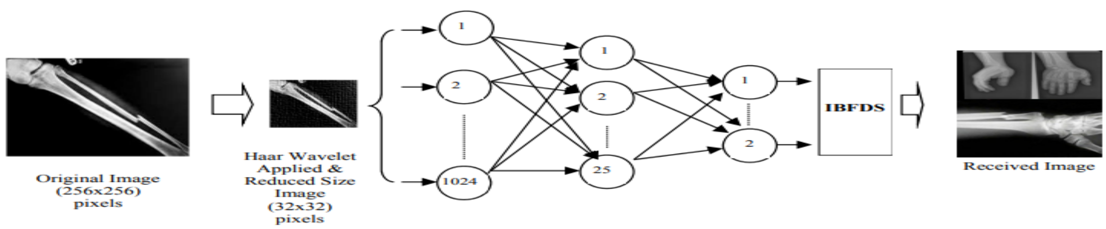


Figure 2 BBNN Architecture

2)Bone Fracture Detection in X-ray Images using Convolutional Neural Network:

The X-ray images are collected from three distinct sources. Dataset is prepared from the X-ray images provided by Vidhya Imaging Centre, Gwalior. A few images are also taken from portable digital X-ray -machines available at the biomedical laboratory in the Electrical Engineering Department of MITS, Gwalior. A dataset of X-ray images has been collected from the Medpix repository, which contains approximately 2000 images approximately. In the experiment initially, 200 X-ray images were chosen, in which 60 images are fractured images, and 140 images are non-fractured.

In this experiment, 80% of the data is given for training & the remaining 20% is for testing purposes.

The proposed method for automated diagnosis of bone fracture detection is prepared. This method aims to classify an X-ray image into fractured or non-fractured cases, which has two processes. The first process is pre-processing (it includes normalization and augmentation). The second process is classification by setting different parameters of the CNN model. The details of pre-processing are explained in the next section. There are a total of seven layers in this CNN model.

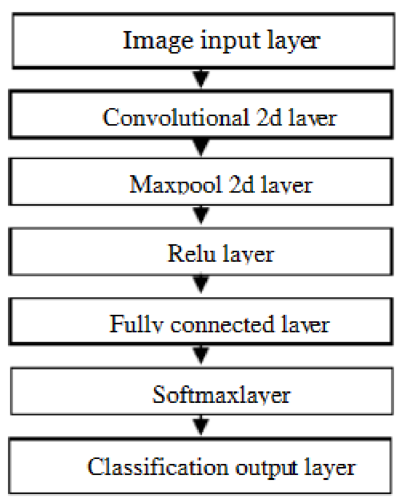


figure 3 CNN model on IBFDS

The paper focuses on classifying X-ray Images, could be able to classify all bone fracture cases with an accuracy of 90% approximately.

CNN Model	Epoch	Optimizer	Batch size	Initial Learning rate	Accuracy (%)	AUC	Specificity
(a)	10	adam	10	1e-05	87.20%	0.8244	87.20%
(b)	20	adam	32	1e-06	76.20%	0.6589	76.20%
(c)	20	adam	32	1e-05	88.00%	0.8286	88.00%
(d)	**20**	**adam**	**32**	**1e-03**	**89.90%**	**0.8088**	**89.90%**
(e)	20	adam	32	1e-04	89.00%	0.8417	89.00%
(f)	20	adam	32	1e-02	86.82%	0.6819	86.82%

Figure 4 IFDS model accuracy

3)Bone Fracture Detection using deep learning (Resnet50)

The data set we used called MURA and included 3 different bone parts, MURA is a dataset of musculoskeletal radiographs and contains 20,335 images described below:

Our data contains about 20,000 x-ray images, including three different types of bones - elbow, hand, and shoulder. After loading all the images into data frames and assigning a label to each image, we split our images into 72% training, 18% validation and 10% test.

The algorithm starts with data augmentation and pre-processing the x-ray images, such as flip horizontal. The second step uses a ResNet50 neural network to classify the type of bone in the image. Once the bone type has been predicted, A specific model will be loaded for that bone type prediction from 3 different types that were each trained to identify a fracture in another bone type and used to detect whether the bone is fractured. This approach utilizes the strong image classification capabilities of ResNet50 to identify the type of bone and then employs a specific model for each bone to determine if there is a fracture present. Utilizing this two-step process, the algorithm can efficiently and accurately analyze x-ray images, helping medical professionals diagnose patients quickly and accurately.

The algorithm can determine whether the prediction should be considered a positive result, indicating that a bone fracture is present, or a negative result, indicating that no bone fracture is present. The results of the bone type classification and bone fracture detection will be displayed to the user in the application, allowing for easy interpretation. This algorithm has the potential to greatly aid medical professionals in detecting bone fractures and improving patient diagnosis and treatment. Its efficient and accurate analysis of x-ray images can speed up the diagnosis process and help patients receive appropriate care.

The accuracy of Elbow Fracture Prediction was 90%, the accuracy of Hand Fracture Prediction was 84%, and the accuracy of Shoulder facture prediction was 82%.

Chapter 3: Methodology

3.1. Datasets

MURA

- MURA is a dataset of musculoskeletal radiographs consisting of 14,863 studies from 12,173 patients, with a total of 40,561 multi-view radiographic images, each belongs to one of seven standard upper extremity radiographic study types: elbow, finger, forearm, hand, humerus, shoulder, and wrist. Each study was manually labeled as normal or abnormal by board-certified radiologists from the Stanford Hospital at the time of clinical radiographic interpretation in the diagnostic radiology environment between 2001 and 2012 To evaluate models and get a robust estimate of radiologist performance, we collected additional labels from six board-certified Stanford radiologists on the test set, consisting of 207 musculoskeletal studies.

GRAZPEDWRI-DX

- GRAZPEDWRI-DX **is** an open dataset containing 20327 annotated pediatric trauma wrist radiograph images of 6091 patients. These annotations include objects and tags of fractures, fracture classifications, and several other relevant image descriptors.

VinDr-SpineXR

- VinDr-SpineXR is a large annotated medical image dataset for spinal lesions, detection and classification from radiographs contains 10,466 spine X-ray images from 5,000 studies, each of which is manually annotated with 7 types of abnormalities by an experienced radiologist.

Fracatlas**:** (For Testing)

- Fractals is a musculoskeletal bone fracture dataset with annotations for deep learning tasks like classification, localization, and segmentation. The dataset contains a total of 4,083 X-Ray images with annotation.

3.2. Use Case Diagram

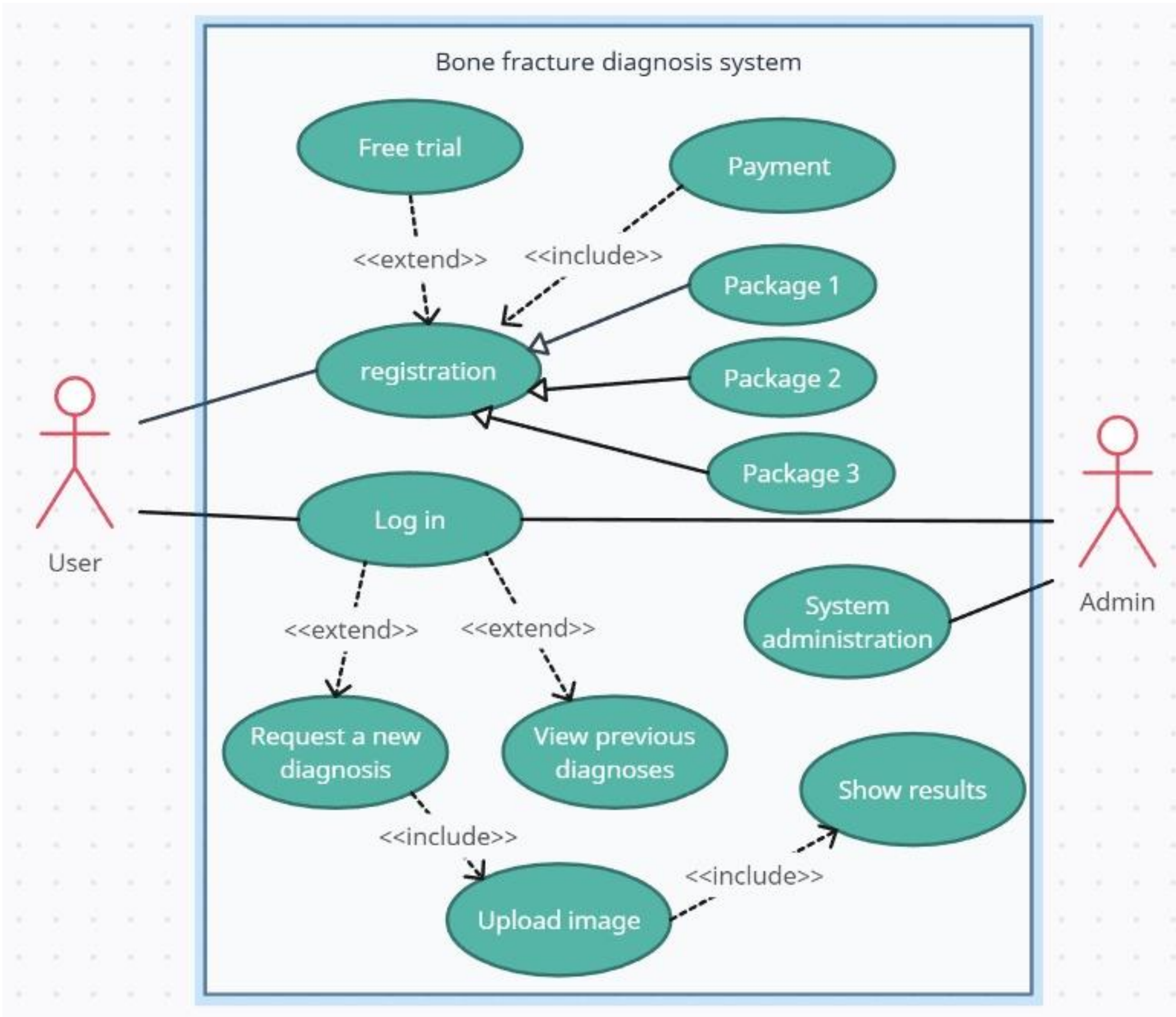


Figure 5 use case diagram

3.3. Sequence Diagram

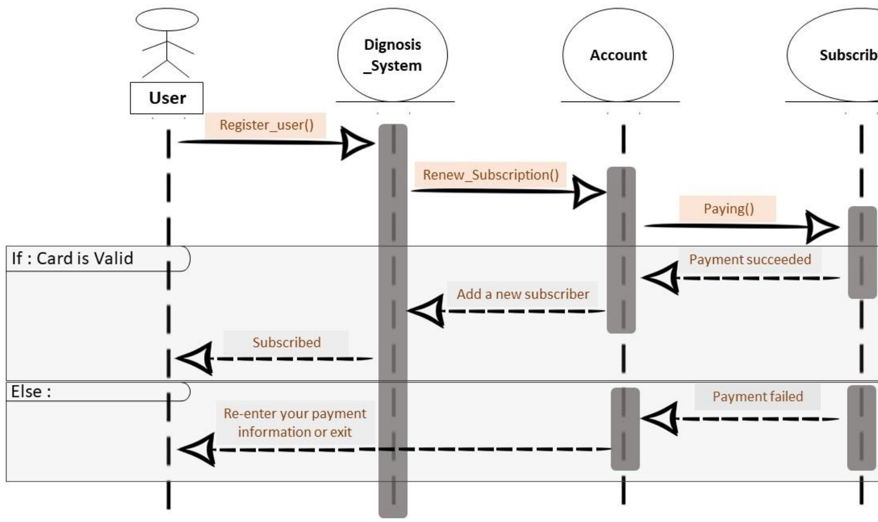


Figure 6 Sequence Diagram

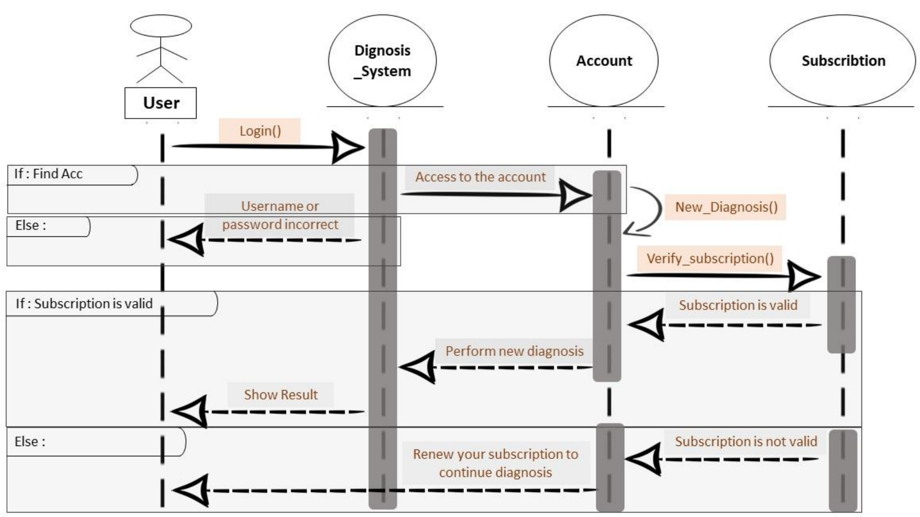


Figure 6 Sequence Diagram

3.4.The User intarface

We have 10 pages of our website:

1. Front End:
   HTML, CSS, JS The core technologies for building the user interface and defining the structure, styling, and interactivity of the dashboard.
   Also,we used Bootsrap library for page's style.
   The Snapshot of our website:
   The sign_up page :user must create an account if he first time visiting our website .
   The sign_in page: if user already have an account ,he make login directly

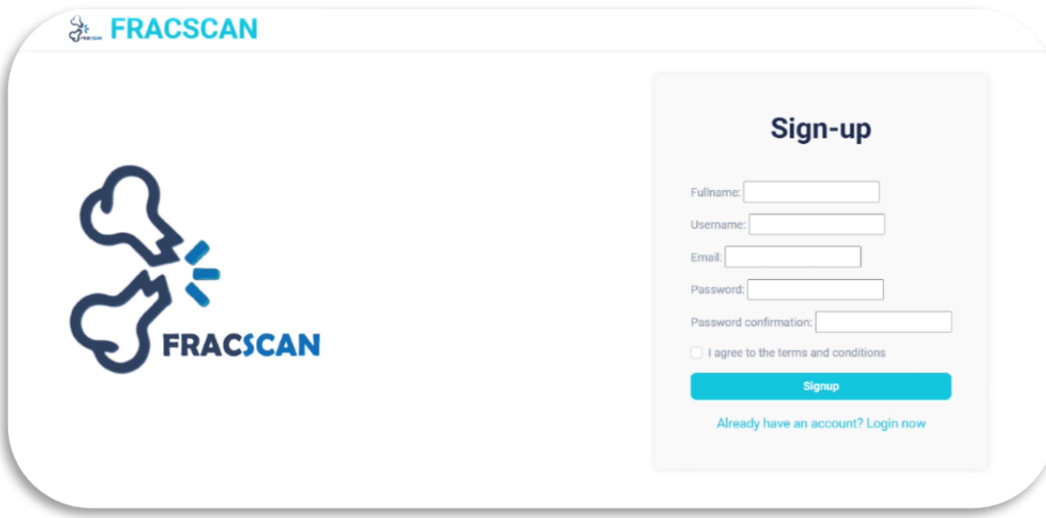


Figure 7 sign up

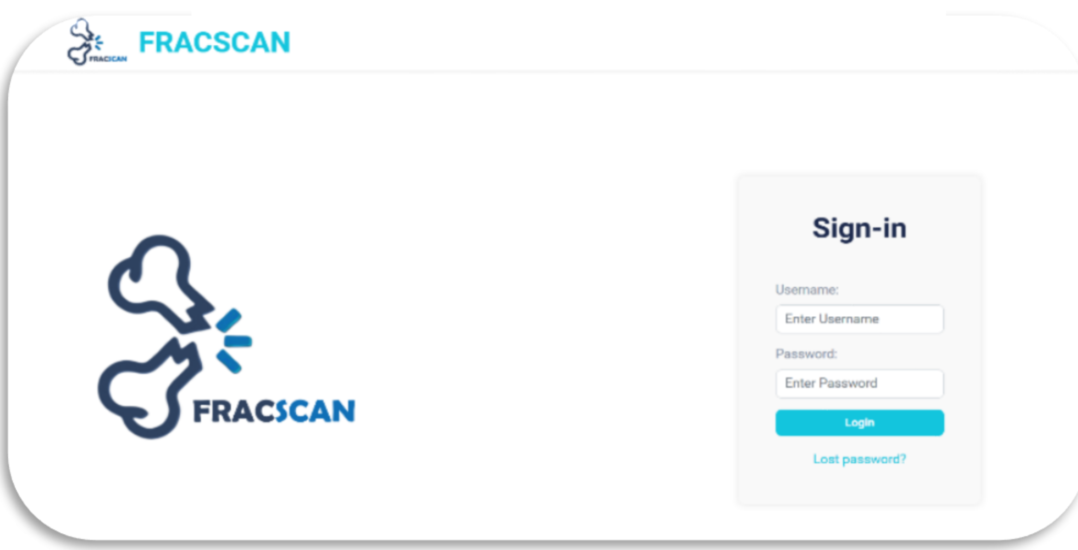


Figure 8 login

Home page:

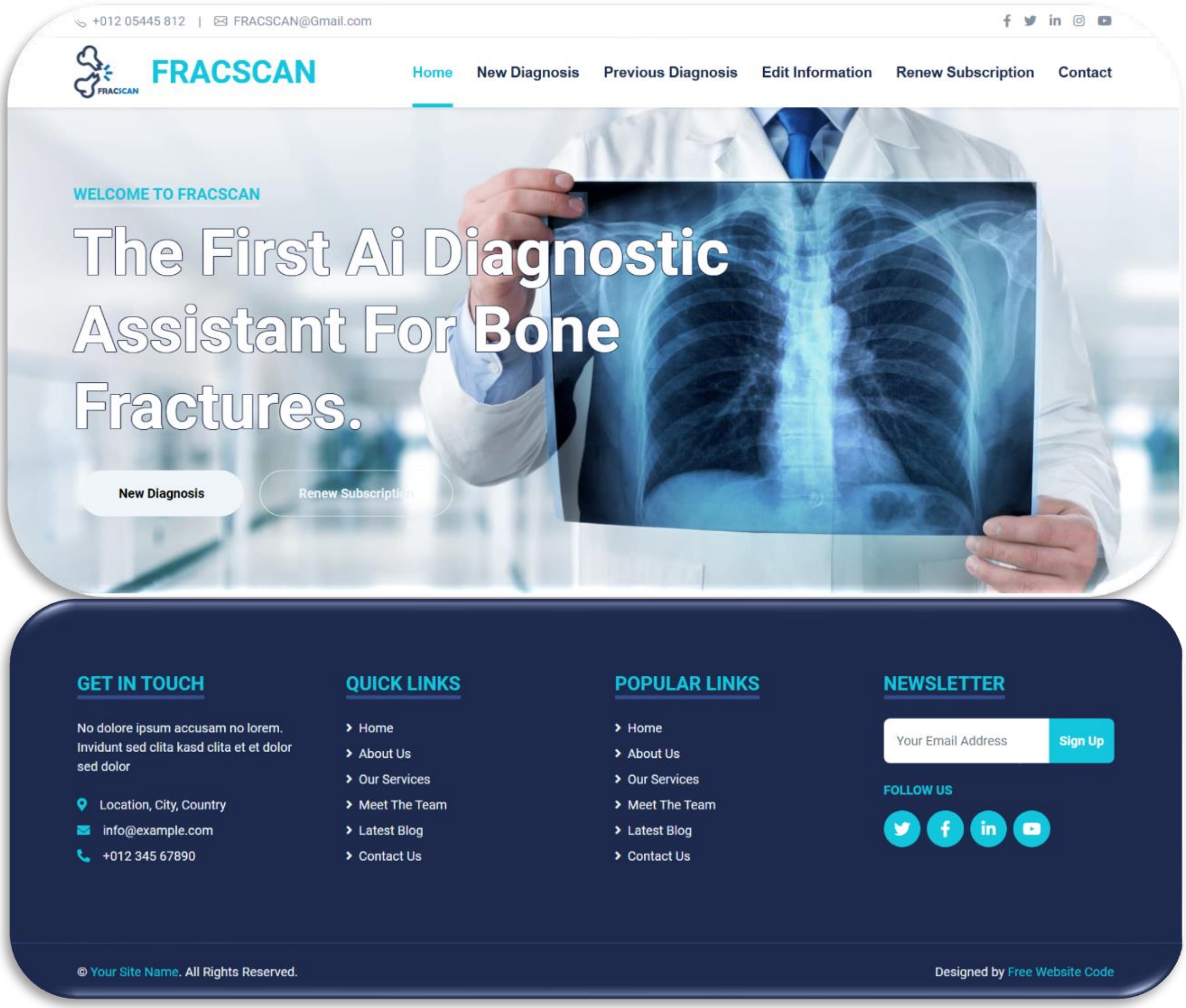
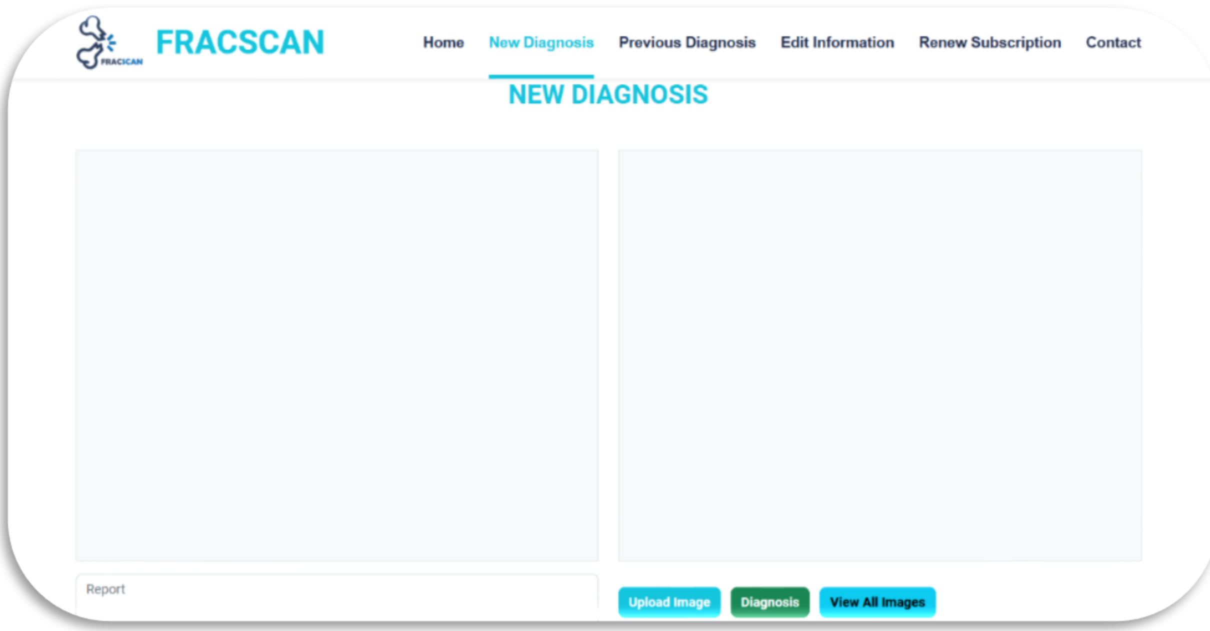


Figure 9 Home page

The New_Diagnosis page : that allows to user upload your X-Ray image, he must upload your image and click on Diagnosis Button.



After uploading X-Ray image:

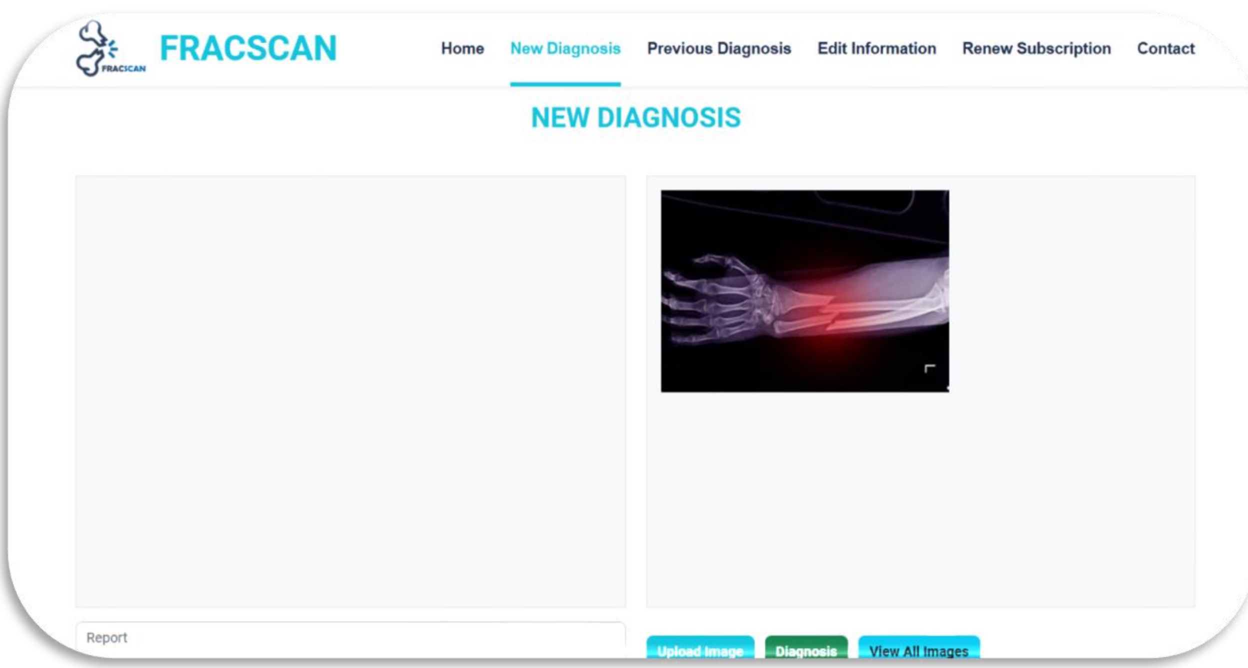


Figure 10 New diagnosis page

The previous Diagnosis page: this pages like the history of usage of user any X-Ray image he upload it in New_Diagnosis page.

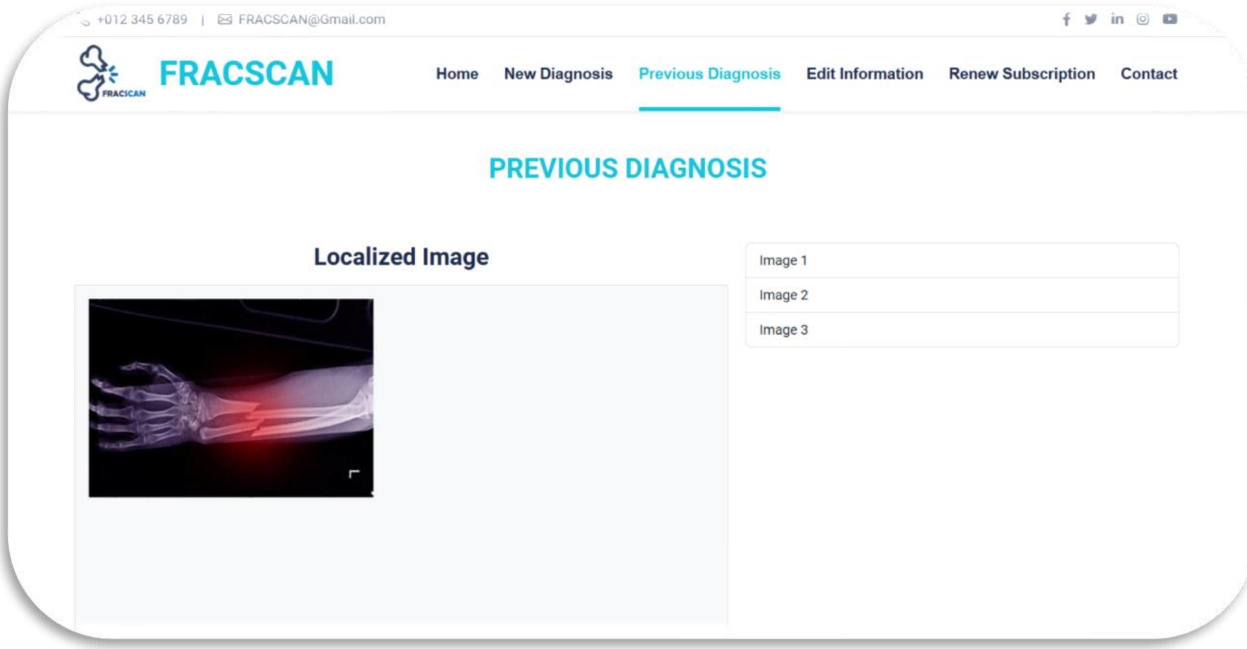


Figure 11 previous diagnosis page

Profile page: contains information about user.

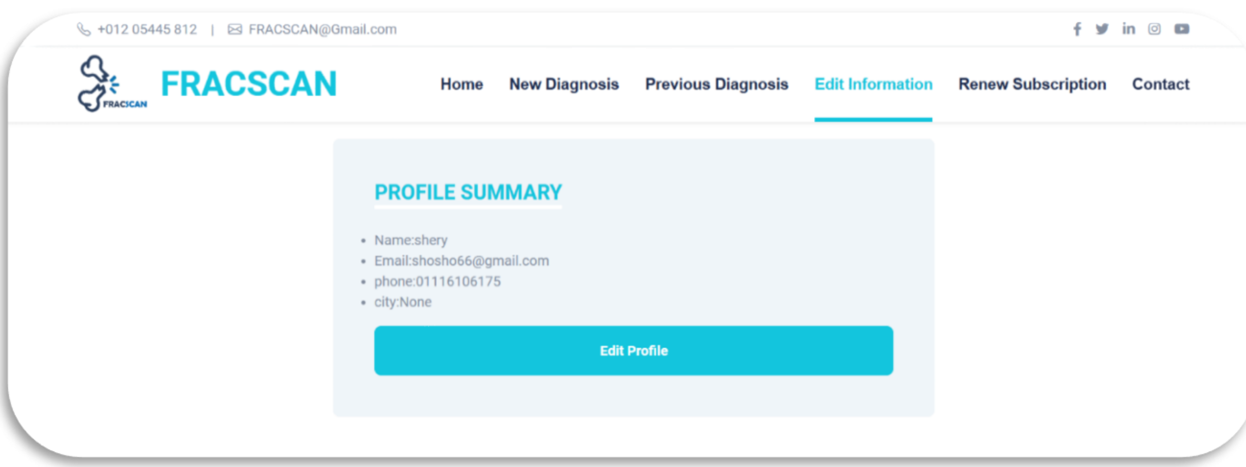


Figure 12 profile page

Edit_profile page: allows users to edit their information such as Email,city,phone_number,..

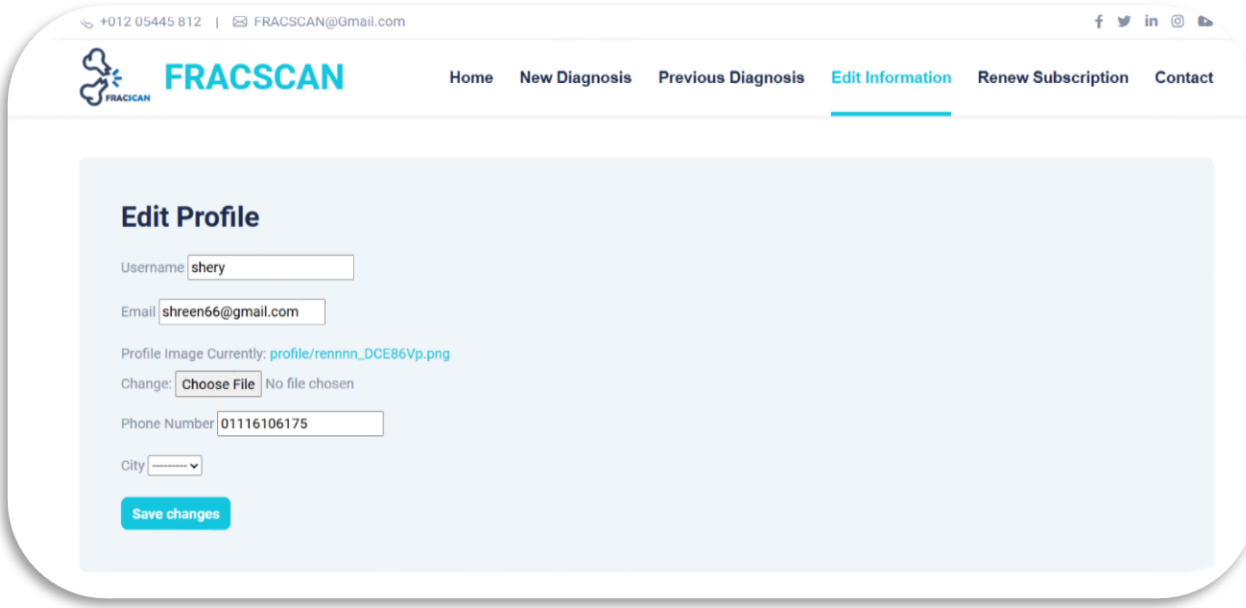


Figure 13 Edit_profile page

Renew subscription page:users must choose in the first which offer that they need to subscribe.

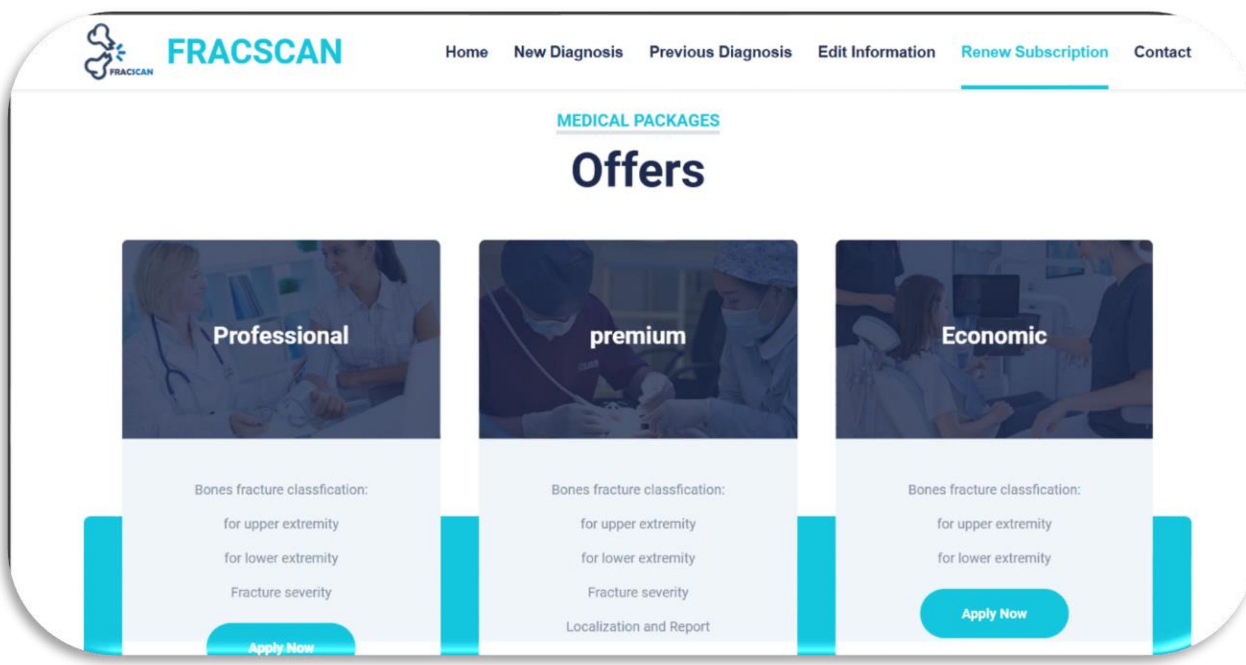


Figure 14 renew subscription page

Payment page: after the user choose the suitable offer ,he must payment to take advantage of the offers

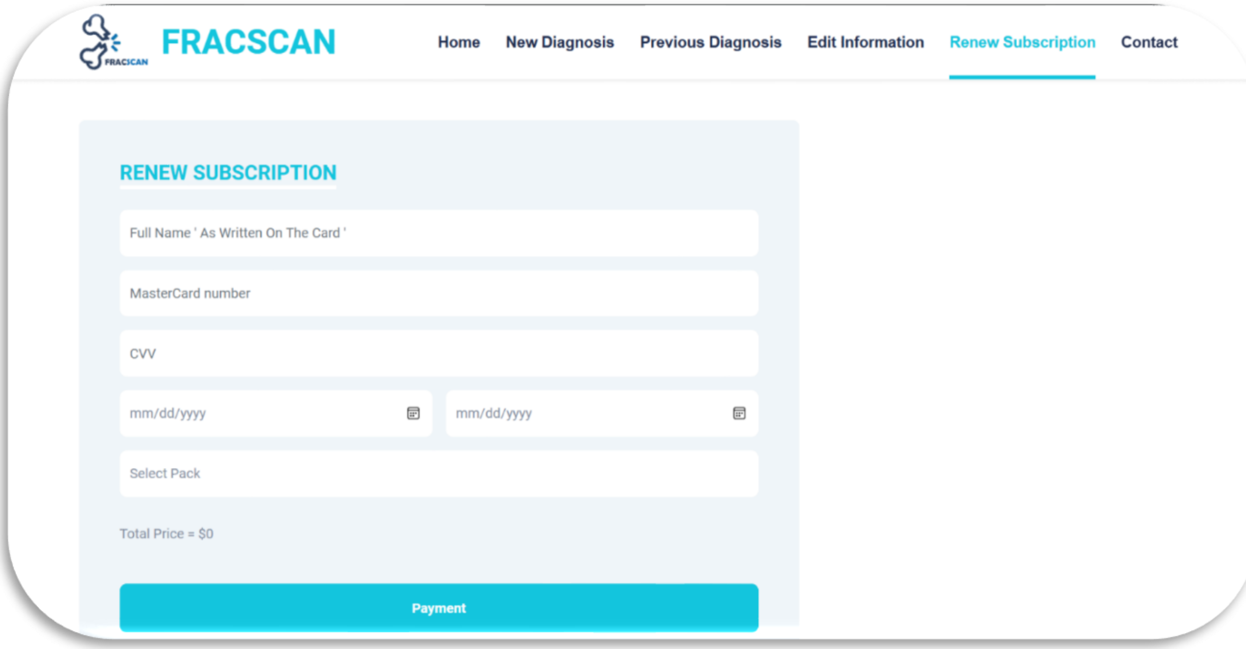


Figure 15 payment page

Contact page: if user need to communicate with our

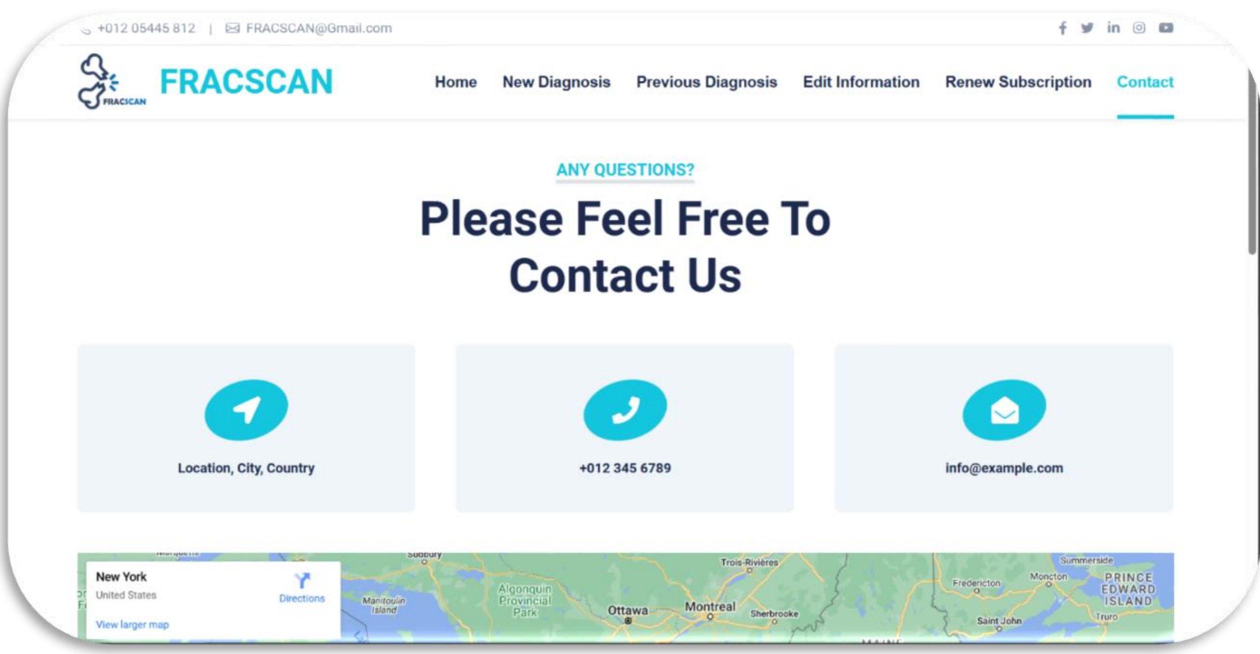


Figure 16 contact page

3.5.Preprocessing:

This stage comprises of the methodology that upgrades the x-ray pictures all together that the ensuing image improves the exhibition of the accompanying phases of the anticipated framework.

In the pre-processing phase, normalization, Contrast Limited Adaptive Histogram Equalization and Data augmentation of a dataset is to be performed.

The first method is the Normalization of data which is used to keep numerical stability in CNN models. The CNN models are used to learn faster and more stable in normalization. The pixel values of the considered images normalize in between the range 0-1. All input images are grayscale images, and rescaling is obtained by multiplying 1/255 with every pixel value for two datasets (mura and Lera)

The second method is Contrast Limited Adaptive Histogram Equalization: in dataset mura the pictures were blurred so, we used this method to clarify the pictures and their lighting, it is a technique used in image processing to improve the contrast of an image while preserving local details.

Here is how it works:

1-Histogram Equalization (HE):

Histogram equalization is a technique that spreads out the intensity values of an image to cover the entire dynamic range more uniformly. It redistributes the pixel values in such a way that the histogram becomes flat, resulting in enhanced contrast.

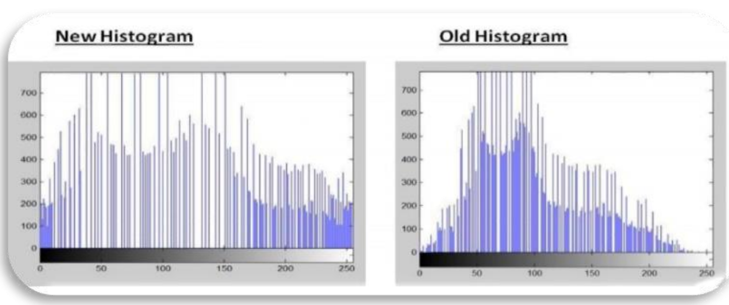


Figure 17 Histogram Equalization (HE)

2-Adaptive Histogram Equalization (AHE):

Adaptive Histogram Equalization (AHE) overcomes the limitations of traditional histogram equalization by dividing the image into small regions and applying histogram equalization independently to each region. This allows for local contrast enhancement, preserving the details in different parts of the image.

3-Contrast Limiting:

To address the overamplification problem of AHE, Contrast Limited Adaptive Histogram Equalization (CLAHE) adds a last step: contrast limiting. This step prevents the overamplification of contrast by clipping the histogram at a specified limit.

We choose the CLAHE prefer to Equalization because The Equalization lighting of the picture is increasing greatly so we use CLAHE to control that with contrast limiting=4 and tile Grid Size= (8,8)

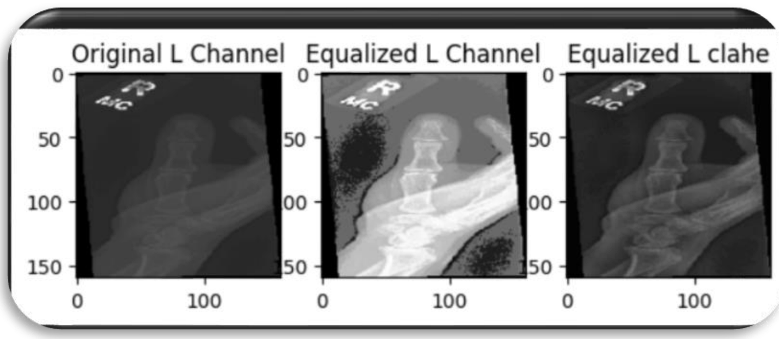


Figure 18 original vs. Equalized vs. Clahe

Data Augmentation on **(Lera dataset after filtering)**, in which CNN models require a massive dataset for practical training. But the existing training images in the input are significantly less (i.e., 600 X-ray images). It is an essential concern when performing X-rays analysis using a deep learning approach while it is tough to gather medical data. To overcome it, the augmentation approach is used to help in increasing the number of images. This also enhances inconsistency in the images and treats them as regularized. The methods used in data augmentation are rotation, flipping, and scaling. The considered X-rays images are augmented with the methods of:

(1) Rotating at the angles of 20 degrees.

(2) Flipping horizontally

(3) width_shift_range = 0.1

(4) height_shift_range = 0.1

(5) Shear_range = 0.2

Data cleaning: When we open Lera dataset, we noticed that for every patient Repeated Images so, we removed those, editing on labels by splitting the data in directories each directory has its bone image.

Data Merging: We faced 2 problems when we tried to merge the 2 datasets "LERA" & "MURA": Size variation LERA has less samples than MURA, so we applied different data augmentations to make them has quite the same number of samples.

Low images contrast We noticed that images are suffering a low contrast problem, so we solved that and figured that this made our images from both datasets follow the same distribution.

3.6.Inception V3

The Inception V3 is a deep learning model based on Convolutional Neural Networks, which is used for image classification. The inception V3 is a superior version of the basic model Inception V1 which was introduced as Google Net in 2014. As the name suggests it was developed by a team at Google
Inception v3 mainly focuses on burning less computational power by modifying the previous Inception architectures.

- Main Architecture:

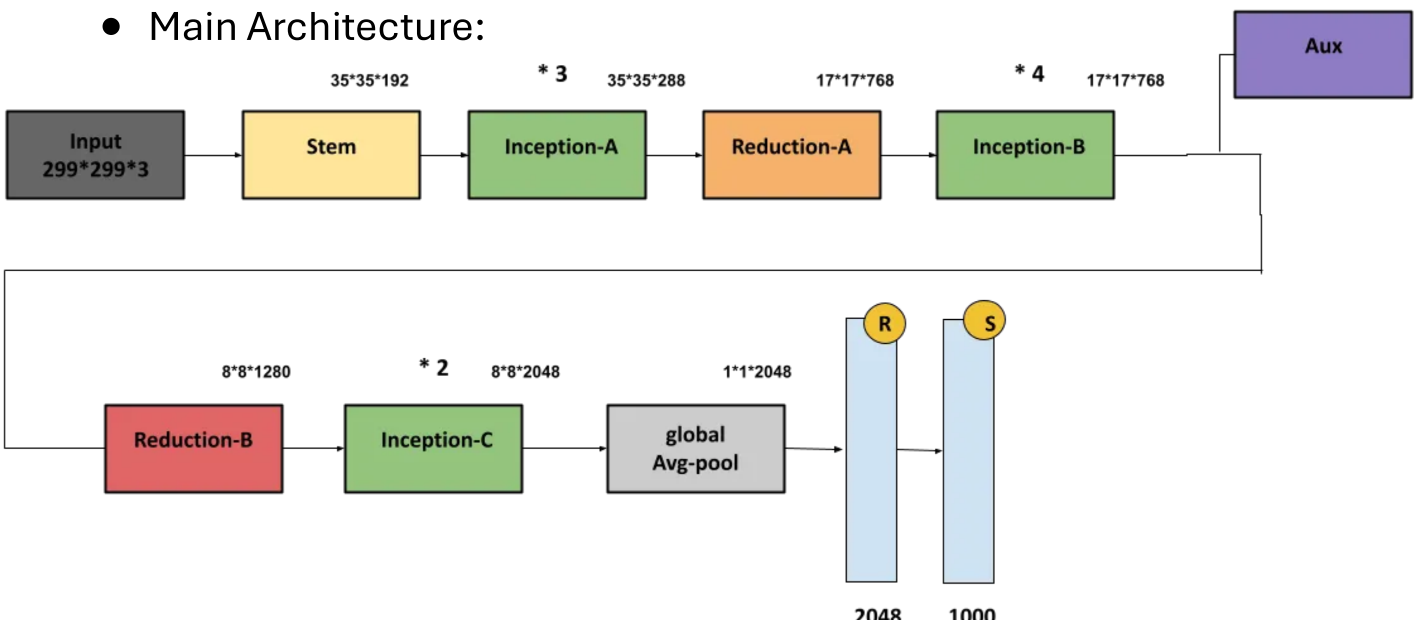


Figure 19 Inception model architecture

Auxiliary classifier: an auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss. In Google Net auxiliary classifiers were used for a deeper network, whereas in Inception v3 an auxiliary classifier acts as a regularizer.

3.7.ResNet-152 - Deep Convolutional Neural Network

ResNet-152 pretrained:

The training of ResNet-152 involves initializing the network's weights and biases, followed by optimizing them using backpropagation and gradient descent-based algorithms. Pretraining on large-scale image classification datasets such as ImageNet is often performed to initialize the network with meaningful feature representations. Fine-tuning can be applied on specific tasks or datasets to adapt the model to the target domain.

ResNet-152 offers several advantages:

Increased Depth: With its even deeper architecture, ResNet-152 can learn more complex representations and handle highly challenging computer vision tasks effectively.

Residual Learning: The use of residual blocks addresses the vanishing gradient problem.

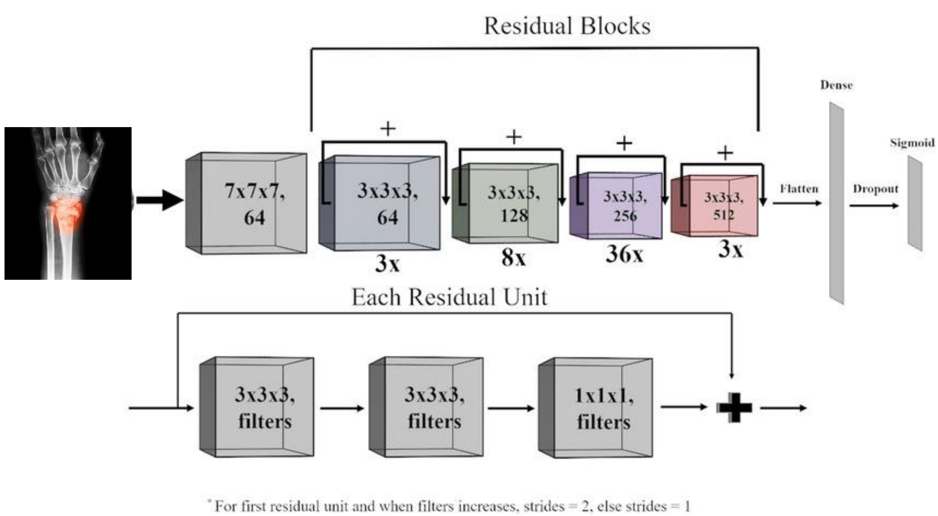


Figure 20 Resnet_152 Architecture

Here's how ResNet-152 works:

ResNet-152 is a deep convolutional neural network, it's designed to address the vanishing gradient problem, It utilizes residual learning by introducing skip connections, These connections allow gradients to flow directly through the network, ResNet-152 consists of 152 layers, making it deep and powerful, It achieves state-of-the-art performance on various image recognition tasks The network architecture enables efficient training

3.8. EfficientNet-B1

Efficient Net is a convolutional neural network built upon a concept called "compound scaling." This concept addresses the longstanding trade-off between model size, accuracy, and computational efficiency. The idea behind compound scaling is to scale three essential dimensions of a neural network: width, depth, and resolution.

Architecture

The first thing is any network is its stem after which all the experimenting with the architecture starts which is common in all the eight models and the final layers [2]. After this each of them contains 7 blocks. These blocks further have a varying number of sub-blocks whose number is increased as we move from EfficientNetB0 to EfficientNetB7 [2].

- Module 1

  — This is used as a starting point for the sub-blocks.
- Module 2

  — This is used as a starting point for the first sub-block of all the 7 main blocks except the 1st one.
- Module 3

  — This is connected as a skip connection to all the sub-blocks.
- Module 4

  — This is used for combining the skip connection in the first sub-blocks.
- Module 5

  — Each sub-block is connected to its previous sub-block in a skip connection, and they are combined using this module.

Depth-wise Convolution is a type of convolution where we apply a single convolutional filter for each input channel. In the regular 2D convolution performed over multiple input channels, the filter is as deep as the input and lets us freely mix channels to generate each element in the output. In contrast, depthwise convolutions keep each channel separate. To summarize the steps, we:

1. Split the input and filter into channels.
2. We convolve each input with the respective filter.
3. We stack the convolved outputs together.

After the depthwise convolution, a pointwise convolution is applied. This is a 1x1 convolution with a set of filters.

- **Efficient Net-B1**

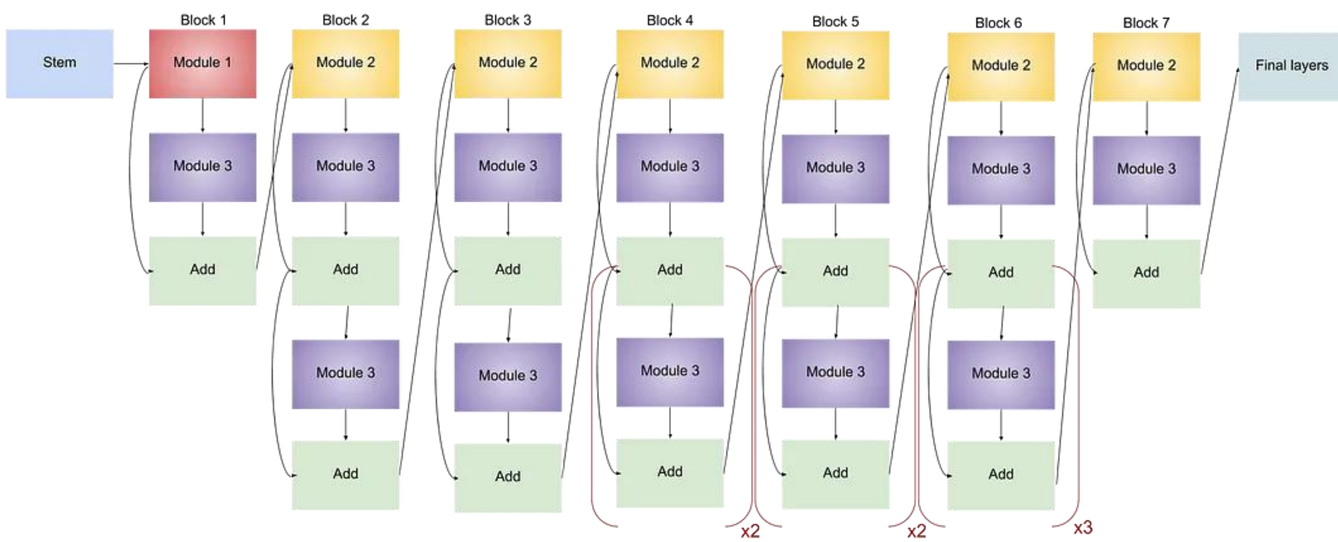


FIGURE 21 Efficient Net - B1 Architecture

3.9. YOLO-V8 L

YOLO (You Only Look Once) is a state-of-the-art, real-time object detection system. YOLOv8 is the latest iteration of the YOLO series, offering significant improvements in accuracy, speed, and flexibility over its predecessors. YOLOv8 builds on the strengths of previous versions while introducing new features and optimizations that make it one of the most powerful object detection models available today.

YOLOv8 Architecture

YOLOv8 introduces several architectural enhancements that distinguish it from its predecessors:

- **Backbone**: The backbone network is responsible for extracting features from the input image. YOLOv8 uses an enhanced version of CSPDarknet, which improves the balance between speed and accuracy.
- **Neck**: The neck of the network helps in aggregating features from different layers. YOLOv8 employs a Path Aggregation Network (PANet) to better fuse features and improve the detection of objects at different scales.

- **Head**: The head of the network is responsible for predicting bounding boxes and class probabilities. YOLOv8 includes advanced detection heads that improve localization accuracy and reduce false positives.

YOLOv8-L: A Specific Variant

YOLOv8-L is a specific variant of the YOLOv8 model, optimized for larger, more complex datasets while maintaining a balance between accuracy and speed. The "L" stands for "Large," indicating that this model version is designed to handle more extensive feature sets and produce higher accuracy for more detailed object detection tasks. Additionally, YOLOv8-L is a versatile model designed not only for object detection but also for classification, posing, and tracking tasks. Its architecture and advanced features enable it to excel in various computer vision applications, making it a powerful tool for multiple tasks in real-time scenarios.

Key Tasks of YOLOv8-L

1. Object Detection

- **Overview**: Object detection involves identifying and localizing objects within an image, providing both bounding boxes and class labels.
- **Features**: YOLOv8-L offers real-time detection with high precision, using advanced detection heads and non-maximum suppression (NMS) to reduce false positives.
- **Applications**: Suitable for surveillance, autonomous vehicles, and robotics.

2. Classification

- **Overview**: Classification assigns a label to an image from a predefined set of categories.
- **Features**: Utilizes the efficient CSPDarknet backbone for high-accuracy classification. Transfer learning can be employed for faster training.
- **Applications**: Medical diagnostics, retail product recognition, and autonomous vehicle classification.

3. Pose Estimation (Posing)

- **Overview**: Pose estimation predicts the positions of key points on objects or humans, such as joints or specific parts.
- **Features**: High accuracy with multi-scale feature aggregation, suitable for detecting key points even in challenging environments.

- **Applications**: Sports analytics, human-computer interaction, and healthcare monitoring.

4. Tracking

- **Overview**: Object tracking follows objects across multiple frames in a video sequence.
- **Features**: Consistent performance with real-time capability, integrates well with tracking algorithms like SORT and DeepSORT.
- **Applications**: Video surveillance, sports analytics, and autonomous vehicle navigation.

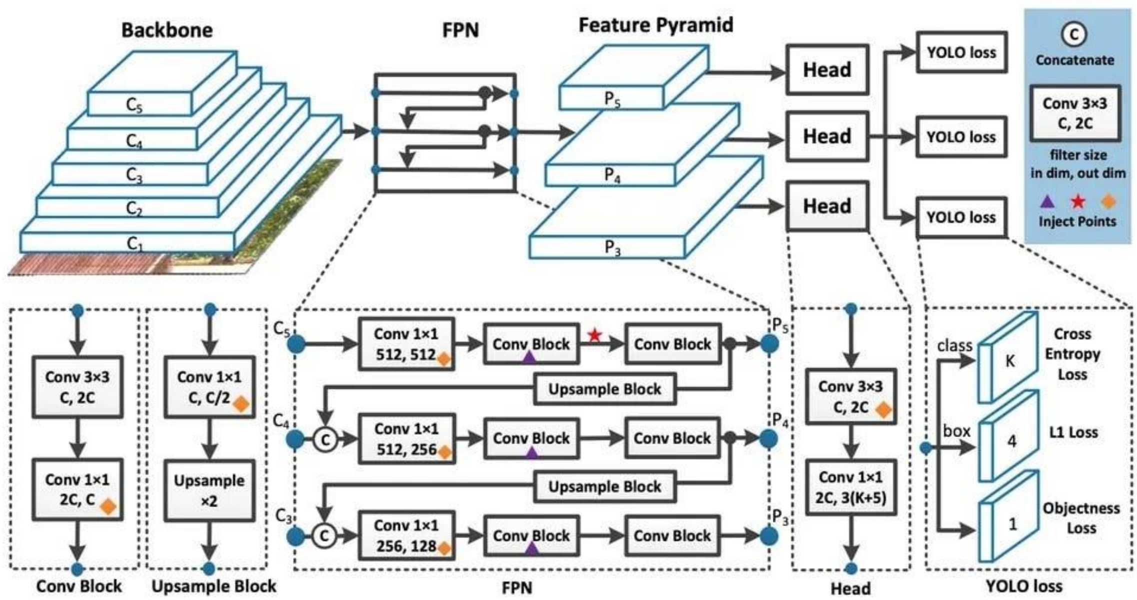


Figure 22 YOLOV8 Architecture

3.10. System Architecture

In FracScan model having 3 models same process and system lifecycle but differ in model architecture, Applying Resnet-152, InceptionV3 and EfficientnetB1 CNN-Based in MURA (upper extremity body) and LERA (lower extremity body) datasets together, explaining in the Block diagram our system architecture but, finally we used efficientnetB5 for the multilabel classification and efficientnetB6 for fracture severity classification.

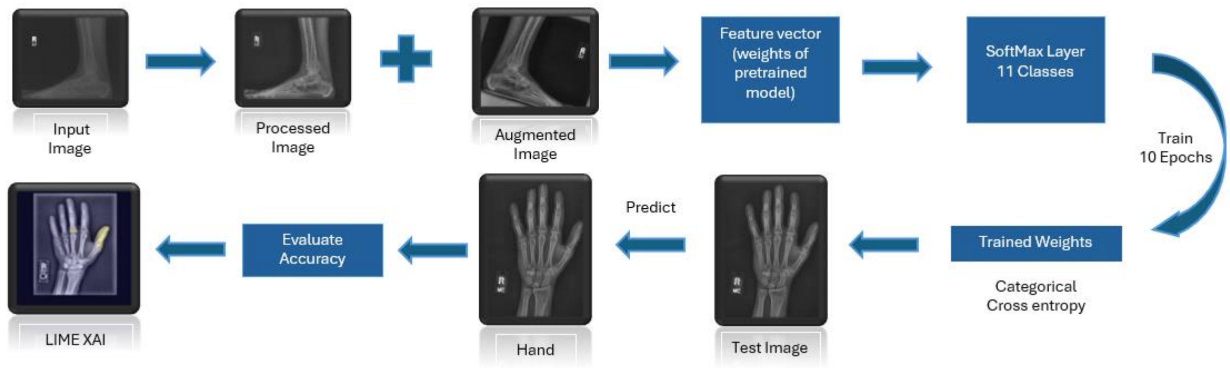


Figure 23 System Architecture

In the advanced phases, which we will explain next, we focus on enhancing our system architecture by integrating the developed web application with the trained models. Each model is assigned a specific task, contributing to a seamless and efficient workflow. This integration allows the system to process inputs and deliver results to users in an organized and accessible manner. By connecting the web application to the trained models, we ensure that the users can easily interact with the system and obtain the desired outcomes. Our goal is to create a robust and user-friendly platform that leverages the power of the trained models to provide accurate and relevant results using APIs.

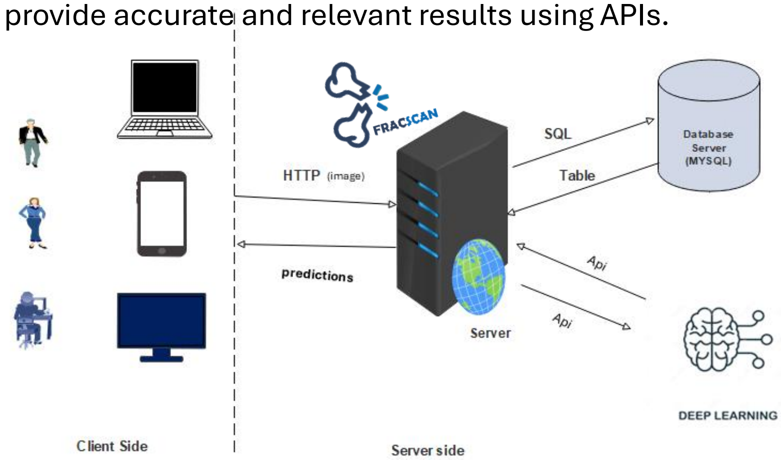


Figure 24 Final System Architecture

3.11. Model of VinDr-SpineXR

We utilize the VinDr-SpineXR Dataset, which comprises a comprehensive collection of 10,466 spine X-ray images. This dataset is meticulously annotated with seven types of abnormalities by an experienced radiologist, ensuring high-quality and reliable labels. Out of the total images, 8,389 are designated for the training set, allowing for extensive model learning and refinement. The remaining 2,077 images are allocated to the test set, providing a robust basis for evaluating the model's performance and generalizability. By leveraging this well-curated dataset, we aim to enhance the accuracy and efficacy of our diagnostic models in identifying and classifying various spinal abnormalities. This dataset's detailed annotations and substantial size make it a valuable resource for advancing our machine learning efforts in medical imaging.

- ## Data Preparation and Cleaning:

  We Convert X-ray images from DICOM files to JPG files to load them using Image Data Generator, remove duplicate images in Excel sheet that contains (Images id , Labels , bounding box annotations) , Handling order of labels in Excel sheet to match with order of X-ray images, we will use YOLO in detection it's required a specific data format to enter to the model which is yaml that accept a path of images directory for train and test the same in labels but with .txt files that determines the class label , x center , y center , width and height that's it yaml format so we should deal with these challenges and put a great data manipulation strategy Firstly we merged all class labels from csv file with boundary box coordinates (xmin,ymin,xmax,ymax) in a 2d array after that coordinates should be like yolo format as I said before so, using these equations xmin+xmax/2 and ymin+ymax/2 for centers and ymax-ymin , xmax-xmin for width and height also normalize width and xcenter by image width and ycenter and height by image height for all data samples and creating .txt file for each sample containing all labels we mentioned and naming this file with the same name as image file for compatibility also coordinating the directories of train's and test's images and labels eventually, creating configuration yolo file .yaml by writing paths of train and test data and writing classes labels, ending with loading ultralytics package to import yolov8s pretrained model and inputting our data.yaml file seeking also for augmentation to our data so defining some parameters in model train for augmentation.

- ## Data Preprocessing:

  We use Contrast Limited Adaptive Histogram Equalization (CLAHE) to Improves the visibility of details in both dark and bright areas of the image (Enhanced Contrast**),** Limits the amplification of noise, unlike traditional histogram equalization (Reduced Noise)

Sample before CLAHE:

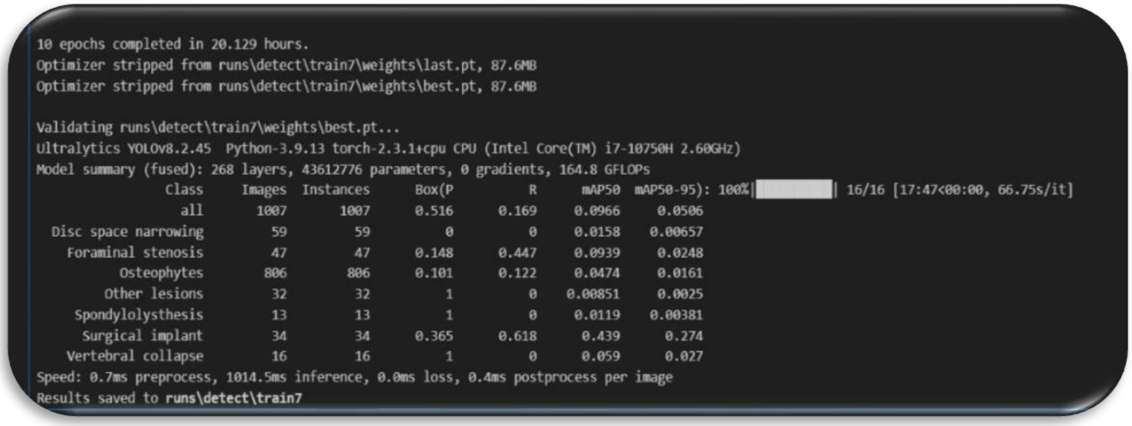
Sample after CLAHE:

- ## Data Augmentation:

After explore Dataset we found class imbalance problem the augmentation approach is used to help in increasing the number of images, The methods used in data augmentation are rotation, flipping, The considered X-rays images are augmented with the methods of: (Rotating at the angles of 30 degrees) , (Flipping horizontally) , (fill mode=nearest) , (Shear range = 0.2) after augmentation number of images in training set become 26,155.

- ## Modeling:

Utilizing YOLOv8 with 20 epochs and a batch size of 32, employing an auto optimizer. The Large version of YOLOv8 necessitates approximately 2 hours per epoch on low computation power devices. Challenges have arisen due to poor data quality and complexity, resulting in lower-than-expected results. Future endeavors will focus on optimizing computational efficiency to overcome these challenges.



```
10 epochs completed in 20.129 hours.
Optimizer stripped from runs\detect\train7\weights\last.pt, 87.6MB
Optimizer stripped from runs\detect\train7\weights\best.pt, 87.6MB

Validating runs\detect\train7\weights\best.pt...
Ultralytics YOLOv8.2.45  Python-3.9.13 torch-2.3.1+cpu CPU (Intel Core(TM) i7-10750H 2.60GHz)
Model summary (fused): 268 layers, 43612776 parameters, 0 gradients, 164.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 16/16 [17:47<00:00, 66.75s/it]
                   all       1007       1007      0.516      0.169     0.0966     0.0506
   Disc space narrowing         59         59          0          0     0.0158    0.00657
     Foraminal stenosis         47         47      0.148      0.447     0.0939     0.0248
            Osteophytes        806        806      0.101      0.122     0.0474     0.0161
          Other lesions         32         32          1          0    0.00851     0.0025
      Spondylolysthesis         13         13          1          0     0.0119    0.00381
       Surgical implant         34         34      0.365      0.618      0.439      0.274
     Vertebral collapse         16         16          1          0      0.059      0.027
Speed: 0.7ms preprocess, 1014.5ms inference, 0.0ms loss, 0.4ms postprocess per image
Results saved to runs\detect\train7
```

Figure 25 Results of YOLOv8

We used InceptionV3 too and add for that GlobalAveragePolling2D Layer and softmax Layer with 20 epochs , a batch size 32 , categorical Cross entropy loss function And Adam optimizer the results were good in the training, but disappointing in the test Accuracy in training = 97.5 , in test = 65.5

3.12. XAI (Explainable AI)

Explainable AI (XAI) refers to the set of techniques and methodologies designed to make artificial intelligence systems more transparent and understandable to humans. The goal of XAI is to provide insights into how AI models make decisions or predictions, thus increasing trust, accountability, and comprehension of AI systems. This is particularly important in scenarios where AI is utilized in critical decision-making processes, such as healthcare, finance, and criminal justice so, we used a guaranteed model called LIME (Local Interpretable Model-agnostic Explanation).

Overview: LIME is a technique used to explain the predictions of machine learning models by approximating the model's behavior in the local vicinity of a specific instance. It provides human-interpretable explanations by fitting a simple, interpretable model (e.g., linear model) to the predictions made by the underlying complex model.

Create LIME Explainer: Instantiate a LIME explainer object, specifying the appropriate parameters such as the number of features (num_features) and the number of samples (Num _samples).

Generate Explanations: Explain the predictions of your machine learning model using LIME. Provide the input image or instance for which you want to generate explanations, along with the model's predict function.

Figure 26 Hand with Most Features help to predict.

3.13. Model of Bone fracture type Classification:

We use Bone Break Classification Image Dataset contains 1129 spine X-ray images (1017 for Training set), (112 for Test set) annotated with10 classes.
The classes are Avulsion fracture, Comminuted fracture, Fracture Dislocation, Greenstick fracture, Hairline Fracture, Impacted fracture, Longitudinal fracture, Oblique fracture, Pathological fracture, Spiral Fracture.

- **Data Preparation:**
  We use a train generator for loading the images of x-rays for each class and split the data in train and validation. We use image size (256,256) and normalize the pixels of them. Then prepare the labeling of them by each image in each class give it label with the name of the class.
- **Modeling:**
  We use the pretrain model efficientb6, batch normalization layer ,dense layer with 32 neurons with active function relu , batch normalization layer, dropout layer and output layer with 10 neurons the number of classes the activation function is SoftMax .The optimizer is Adam with learning rate0.0001, loss is categorical cross entropy. We get accuracy in training is 96.76% and accuracy in testing is 96.43% after 40 epochs

3.14. Model for GRAZPEDWRI-DX

GRAZPEDWRI-DX **is** an open dataset containing 20327 annotated pediatric trauma wrist radiograph images of 6091 patients. These annotations include objects and tags of fractures, fracture classifications, and several other relevant image descriptors.

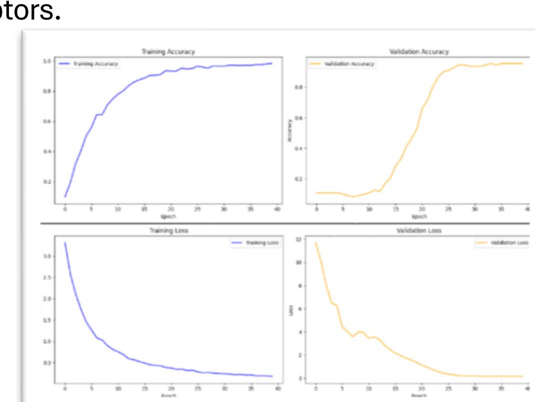


Figure 27 learing curves fracture type

3.14.1) preprocessing

We found resolution of dataset photo not fixed so we were not have the ability to use normalization direct so we iterate for every photo on the dataset and take her size and normalize it one by one

We remove all photo that under10 years because it will cause confuse to the model

3.14.2) prepare for yolo

We use yolo v8l model for this dataset before that yolo need some preparation to work

3.14.2.1) Text file for every label row

Yolo model need for every label row text file indicate the label class and the coordination for the photo indicate xmin, ymin , xmin and xmax we do ir manual we iterate for every row we take it class and coordination and put them on separate file

3.14.2.2) data.yaml file

Yolo need yaml file contain train data set path and test data set path and the Text file for every label row as we mentioned before and class representation name for every class on the data

3.14.3) yolov8

YOLOv8 continues the YOLO (you only look once) tradition of providing fast and accurate object detection, with significant improvements in architecture, training efficiency, and edge device performance. Its integration with the latest deep learning frameworks ensures it stays at the forefront of object detection technology.

We use yolo model version 8 and large version we train on 50 epochs we chose 640,640 image size and we choose 8 batch size and make the optimizer default from the model and for more speed we use coda to run the model with both CPU and GPU.

3.14.4) model evaluation

Using a configuration of 50 epochs and automatic batch sizing, GPU processing with image dimensions set at 640x640 was applied to YOLOv8-L for object detection. Achieving peak performance with 58 mAP for optimal weights and 48 mAP for the range 50-99, accompanied by favorable classification and bounding box losses, subsequent predictions demonstrated highly accurate results, as illustrated in the following figure.

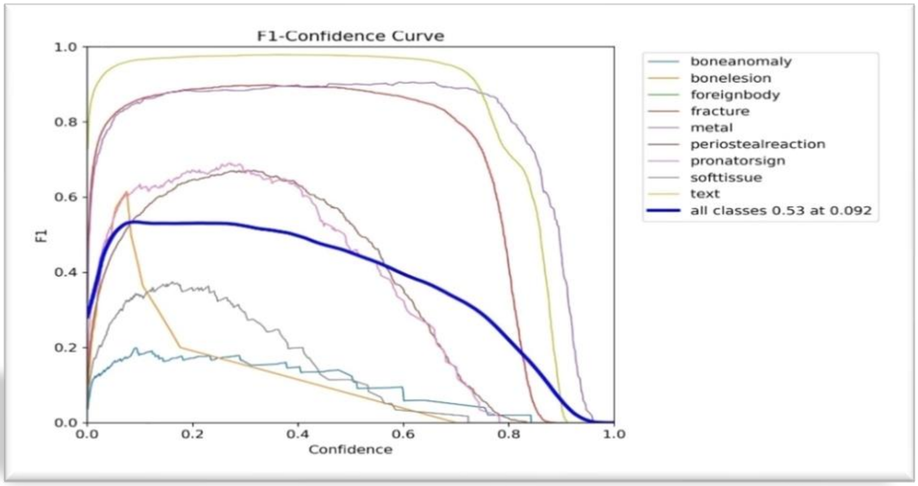
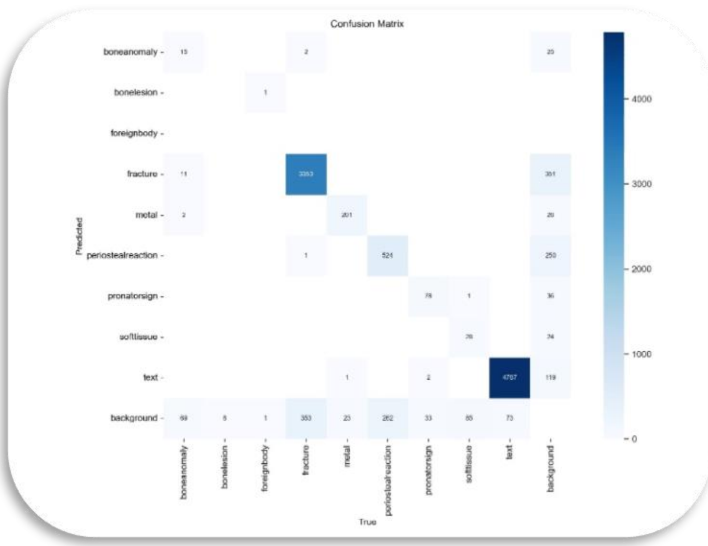


Figure 28 f1-confidence curve



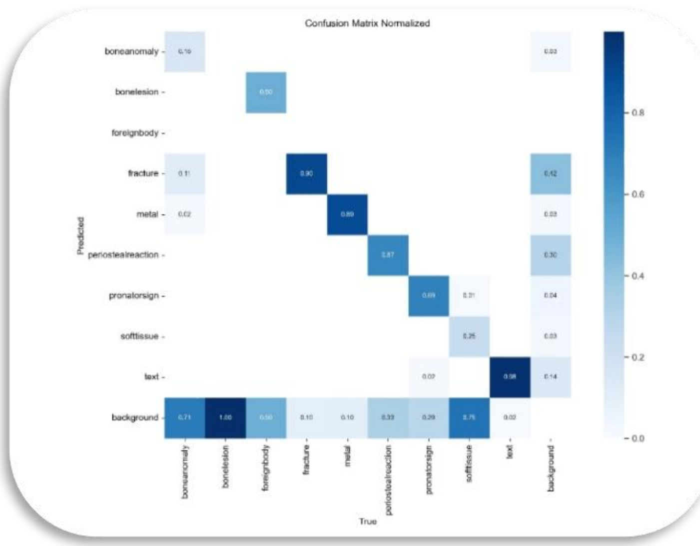


Figure 29 confusion matrix

Figure 30 confusion matrix normalized

Figure 31 Wrist Fracture Detection

3.15. Multi-label Classification (Upper and lower extremities)

Overview

FracScan is a comprehensive image processing system designed to enhance the accuracy and efficiency of bone fracture detection and bone type classification. By merging datasets and optimizing model architecture, FracScan aims to reduce trainable parameters while providing a unified solution for users.

Dataset Integration

FracScan utilizes a merged dataset strategy to bolster model robustness and streamline user interaction. By combining disparate datasets, the system ensures a more comprehensive training regimen without requiring users to select between different models, FracScan merges datasets to enhance model robustness. It combines fracture class images with bone type classification, facilitating a unified training approach. Images are resized uniformly to 256x256 pixels for consistency.

Multilabel Classification Approach

A key innovation in FracScan is the adoption of multilabel classification for bone type and bone fracture identification. This approach optimizes resource utilization and operational efficiency by enabling simultaneous classification tasks within a single model framework.

Implementation Details

1. **Dataset Preparation**:
    - Augmentation: FracScan augments fracture class images to enhance model training diversity.
    - Image Path Handling: Image paths are sourced from CSV files to identify 'positive' scans based on predefined criteria from the Mura and Lera datasets.
    - Each bone type (e.g., ANKLE, ELBOW) has a dedicated directory structure, simplifying image labeling.
2. **Multilabel Classification**:
    - Classes: FracScan categorizes images into distinct bone types: ANKLE, ELBOW, FINGER, FOOT, FOREARM, HAND, HIP, HUMERUS, KNEE, SHOULDER, WRIST.
    - Data Generation: Images and corresponding labels are processed using ImageDataGenerator, ensuring batched data handling with shuffle capability.

3. **Data Labeling**:
    - Mura Dataset: Regular expressions are employed to identify positive scans based on specific keywords related to fractures.
    - Lera Dataset: Each scan type is listed with corresponding patient study numbers marked as positive or negative, ensuring precise data labeling.

4. **Data Dimensions**:
    - Training Data: A 2D array is initialized with dimensions (64641, 1) to accommodate training samples, combining fracture and bone type classification.
    - Testing (Validation) Data: Similarly, a 2D array with dimensions (3371, 1) is prepared for testing/validation, ensuring robust model evaluation.

5. **Integration Challenges:**
    - Data Merging: Challenges arise in synchronizing fracture labels with bone type labels during training due to the large training set size (64643 samples).
    - Custom Generator: Developed to synchronize shuffled indices from the ImageDataGenerator with fracture labels, ensuring alignment between bone type and fracture classification for each sample.
    - Batch Concatenation: Implements custom indexing to concatenate bone type and fracture labels seamlessly within each batch, maintaining data integrity.

6. **Data Generator Implementation**:
    - **Training Setup**: Utilizes ImageDataGenerator with a batch size of 32 and shuffling to enhance training diversity and effectiveness.
    - **Label Format**: Each batch yields labels with dimensions (32, 12), reflecting 11 bone types and 1 bone fracture.

- **Image resizing**: Each image is resized to 256x256 pixels for consistency.

But, we faced these challenges successfully and passes this custom data to the efficientb5 pretrained using tensor hub but the trainable parameters are freeze cause of high computation power needed we not have, we add 6 dense layers with 32 neurons , 6 batch normalization and 6 dropout to ensure hard training and prevent fall in overfitting the output layer actually is sigmoid because it's multilabel classification so we need more than one class predicted and the loss function is focal loss to concentrate on last class which is abnormality class so after 15 epoch in binary cross entropy loss and 5 in focal loss with gamma = 2 and alpha = 0.25 we get 94 accuracy on training and 93,6 on validation but we plot the training of the last 5 epochs which in focal loss and plot the learning curves.

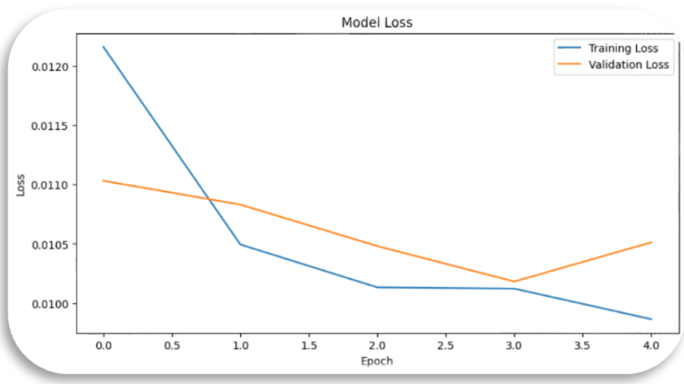


Figure 32 loss over 5 epochs

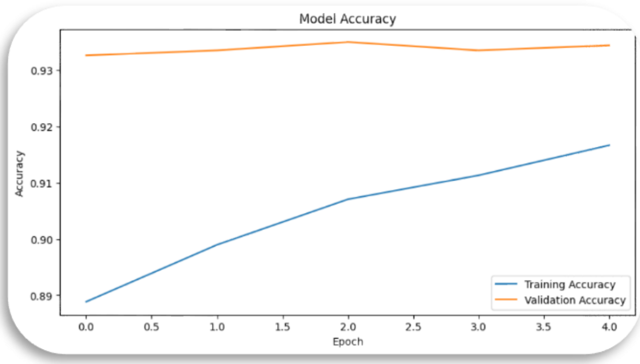


Figure 33 accuracy over 5 epochs

Post-Training Evaluation

After training, FracScan applies thresholds to predicted labels for optimal classification results:

- **Thresholds**: Default threshold of 0.5 for bone type classification and 0.3 for abnormality class to maximize recall.
- **Metrics**: Computes F1-score, precision, and recall for each class. Precision in the abnormality class is noted at 50%, indicating room for improvement.
- **Future Directions**: Plans to enhance precision through additional well-annotated data training to refine abnormality detection capabilities.

FracScan continues to evolve in enhancing diagnostic accuracy in clinical settings by leveraging advanced model training and evaluation strategies, for more detailed providing metrics and confusion plots that describes precision and recall with colors Lighter Shades of blue usually represent lower counts or percentages of predictions but, darker Shades of blue Indicate higher counts or percentages of predictions and White**:** Typically used for cells with fewer instances or lower courts across the matrix.

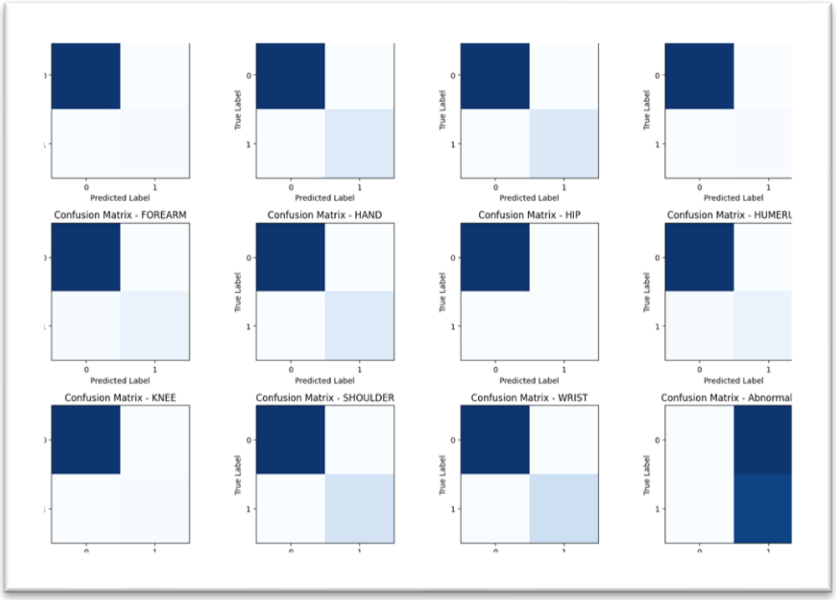


Figure 34 Confusion for classes

Also, we have F1-score, precision, recall and Maps metrics to show and differentiate between classes Eventually, we test on an image providing lime which we used before for bone type explanations but this time for abnormality label explanations it compensates for localization annotations which of the data lack.

Figure 35 Lime positive shoulder

	precision	recall	f1-score	support
0	0.92	0.84	0.88	43
1	0.93	0.93	0.93	465
2	0.98	0.95	0.97	461
3	0.89	0.91	0.90	45
4	0.93	0.81	0.87	301
5	0.96	0.95	0.96	460
6	0.95	0.64	0.77	28
7	0.98	0.78	0.87	288
8	0.98	0.86	0.92	58
9	0.97	0.99	0.98	563
10	0.98	0.95	0.96	659
11	0.49	0.99	0.65	1635
micro avg	0.72	0.95	0.82	5006
macro avg	0.91	0.89	0.89	5006
weighted avg	0.81	0.95	0.85	5006
samples avg	0.72	0.94	0.79	5006

Figure 36 Result Metrics

3.16. NLP Report using LSTM

Text generation models are designed to predict and generate subsequent words in a sequence, creating coherent and contextually relevant text. This guide focuses on using an LSTM-based model due to its effectiveness in handling sequential data and learning long-term dependencies, which are essential for generating meaningful text, Fracscan tried to have real documented reports from physicians with images to train to transformers-based models like bert but, it’s not available now seeking out to having it in the future.

Data Preparation

To build an effective text generation model, it's crucial to start with well-prepared data. The process begins by gathering a collection of documents that the model will be trained on. Each document should be a coherent piece of text, typically related to a specific subject or theme. Once the documents are collected, they are tokenized, converting the text into sequences of integers where each integer represents a unique word. This transformation allows the model to process and learn from the text data. Additionally, the sequences are padded to ensure uniform length, facilitating efficient training.

Model Architecture

The architecture of the LSTM-based text generation model is designed to capture and utilize the sequential nature of text data. It typically starts with an embedding layer, which transforms the integer-encoded words into dense vectors of fixed size, providing a meaningful representation of the words. This is followed by one or more LSTM layers that process these embeddings. The LSTM layers are adept at learning and remembering long-term dependencies in the text, making them ideal for generating contextually accurate text. Finally, a dense output layer with a softmax activation function is used to predict the next word in the sequence, outputting probabilities for each word in the vocabulary.

Training the Model

Training the LSTM-based text generation model involves feeding it the prepared sequences of text data. The model learns to predict the next word in a sequence based on the preceding words through a process called supervised learning. During training, the model's predictions are compared to the actual next words in the sequences, and the difference (loss) is used to adjust the model's weights. This iterative process, known as backpropagation, continues for a specified number of epochs, gradually improving the model's accuracy. The training process requires substantial computational resources and time, especially for larger datasets so, we can generate text by seeding a word to the trained model and determine how many words to generate.

Chapter 4: Deployment

4.1. API

API Design and Implementation

The core functionality of the FracScan application is to analyze X-ray images for bone fractures using a deep learning AI model. This functionality is exposed through a RESTful API, which is implemented using the Flask framework in Python. The API handles the process of receiving an image from the user, processing it through the AI model, and returning the diagnosis results.

API Overview

The FracScan API consists of several endpoints, each designed to handle specific tasks within the image diagnosis workflow. The primary endpoint is responsible for receiving the X-ray image, processing it, and returning the results of the diagnosis, including the presence of fractures, their types, and their locations.

Endpoint:

- Description**:** This endpoint receives an X-ray image uploaded by the user, processes it using the deep learning AI model, and returns a diagnosis report.
- Request Parameters**:**
    - file: The X-ray image file to be analyzed. This should be uploaded as jpg
- Response:
    - data: An object containing the diagnosis results. If the image is successfully processed, this object includes:
        - fractured: A Boolean indicating whether a fracture is detected.
        - fracture type: A string describing the type of fracture (e.g., "comminuted", "greenstick").
        - location: An array of coordinates marking the location of the fracture on the image.

Implementation Details

The implementation involves several key steps:

1. **Image Upload and Validation:** The uploaded image is received and validated to ensure it is in the correct format. The validation process also checks for common issues such as missing files or incorrect content types.
2. **Image Preprocessing:** The uploaded image is preprocessed to meet the input requirements of the AI model. This may include resizing, normalization, and other transformations.
3. **AI Model Inference:** The preprocessed image is fed into the deep learning AI model to generate a diagnosis. The model outputs predictions indicating the presence and type of fractures, as well as the location of fractures on the image.
4. **Response Generation:** Based on the model's predictions, a response is generated and returned to the user. The response includes the diagnosis results formatted in a JSON structure.

4.2. Backend

We used Django ,is a high-level Python web framework that encourages rapid development and clean, pragmatic design.

Key Features:

1_ORM (Object-Relational Mapping):

- Allows you to interact with your database, such as creating, retrieving, updating, and deleting records, without writing raw SQL queries.

2_Admin Interface:

- Automatically generates a web-based administrative interface for your models, making it easy to manage your application’s data.

3_Templates:

- Django's template system allows you to create dynamic HTML pages using Django Template Language (DTL).

4_URL Routing:

- Maps URLs to views. Django uses a URL dispatcher that allows you to design URLs for your application.

5_Views:

- Functions or classes that receive web requests and return web responses. Views are responsible for executing business logic and rendering templates.

6_Forms:

- Django forms handle HTML forms and abstract the process of generating HTML form elements and validating user input.

7_Security:

- Includes features such as SQL injection protection, cross-site request forgery (CSRF) protection, cross-site scripting (XSS) protection, and clickjacking protection.

8_Internationalization:

- Supports translating your application into multiple languages.

Database on our website: Django supports several databases, including SQLite which we used it.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Snapshots of our database:

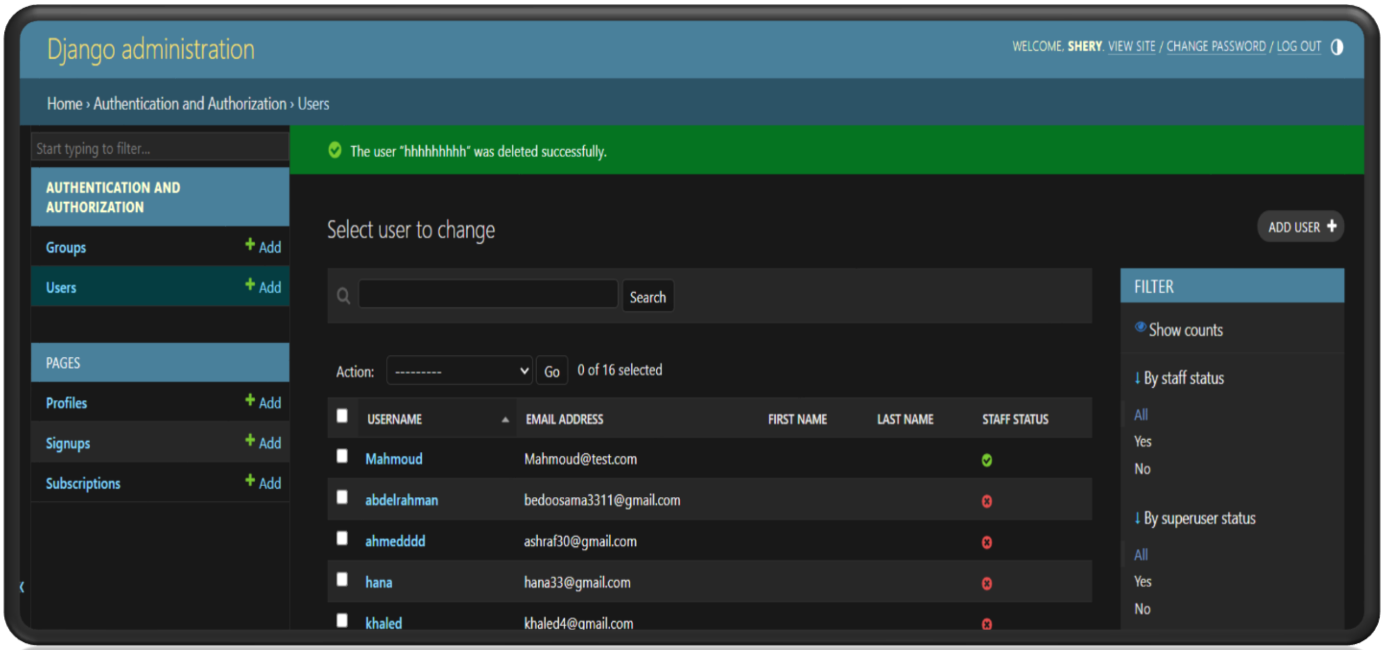


Figure 37 Django database

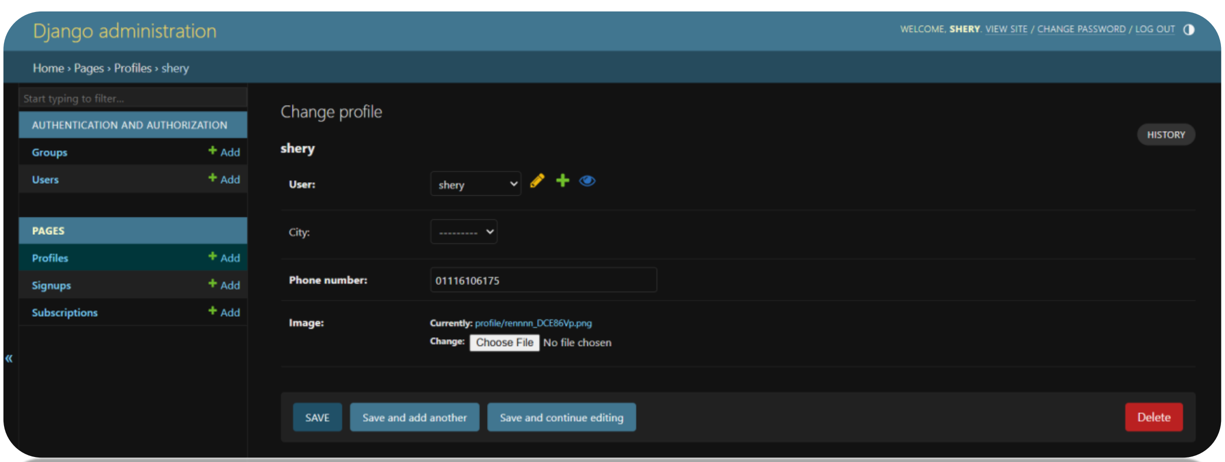


Figure 38 Django Database profile

Chapter 5: Tools & Technologies

5.1. Python-Django

We used python as the main programming language for development Adamdam for its richness of libraries, optimization and simplicity in Ai development also using Django from python for backend to deploy fracscan models to webapp.

5.2. GitHub

Git packages and GitHub make our communication easy for transfer commits and modifications in any project development phases due to its scalability, security and powerful server where we upload the project in its repository.

5.3. Figma

In prototype within deployment phase, we should have a good UI for Fracscan website so, Figma helped us by offering powerful prototyping capabilities allowing us to create interactive prototypes to demonstrate user interaction.

5.4. TensorFlow

Deep Learning and big Neural Networks require a stable, optimized, flexible, high performance and open-source framework like TensorFlow which helped us in several issues related to optimization for our computing resources.

5.5. Keras

Using keras is an integral part of TensorFlow because of its Strong High-Level API Enables users to build and train neural networks using simple, intuitive syntax.

5.6 OpenCV

Before developing some models, we should treat with images doing image preprocessing to simplify feature extraction operation for the model.

5.7. Lime

LIME (Local Interpretable Model-agnostic Explanations) is a technique used in machine learning for explaining the predictions of black-box models. It provides interpretable explanations at the instance level, helping users understand why a model made a particular prediction for a given input data point by highlighting super features in the image.

5.8. Flask

Flask, a lightweight and flexible web framework for Python, is employed in this project to build RESTful APIs. Its simplicity and ease of use make it ideal for handling HTTP requests, managing routes, and integrating with machine learning models. Flask allows for quick development and seamless deployment of web applications, making it a popular choice for backend services.

5.9. YOLO

The project leverages YOLOv8 for object detection and localization, harnessing its advanced capabilities to accurately identify and localize objects within images or video frames. This is crucial for tasks requiring precise spatial information for each detected object.

Chapter 6: Conclusion

6.1. Evaluation:

This project, named FRACSCAN, focuses on using AI models and deep learning algorithms to classify different types of bones with a high percentage of accuracy, represents a significant advancement in medical imaging technology.

Through the amalgamation of diverse datasets encompassing various fracture patterns and anatomical regions, the proposed system learns intricate features and patterns crucial for types of bones classification. Leveraging advanced image processing techniques, such as image enhancement and segmentation, enhances the system's ability to extract pertinent information from radiographic images, facilitating precise analysis and classification.

Here are our evaluation results in all models we have done yet in Fracscan:

Table 1 Evaluation

MODEL	Epoch	Optimizer	Batch size	Dataset	Accuracy
Efficient Net-B5	10	Adam	32	Mura and Lera Upper and lower body - Bone type and abnormality classification	93%
Inception v3	20	Adam	32	Vindr-Xr Spinal bones	67%
Efficient Net-B6	10	Adam	32	Bone fracture type	96%
YOLOV8-L	50	Adam	Auto	GRAZPEDWRI-DX	58 mAP (mean - average precision)
LSTM	50	Adam	32	Websites Reports	98%

6.2) Project Plan:

Table 2 Gantt-Chart

Task	Task title	Description	Task status (completed / Duration)
1	Documentation	Documentation for final discussion	(Completed / two weeks)
2	Deployment of website	User interface for user	(Completed /two month)
3	Preprocessing	Preprocessing the data to use on model	(completed /one month)
4	Model	Implement the models and evaluate	(Completed/one month)

6.3. Future Work

Discussing future goals for Adamdam and the insights that improve productivity to fulfill our audience’s needs so, there are some future works will be achieved in the future Inshallah described level of difficulties on the below figure.

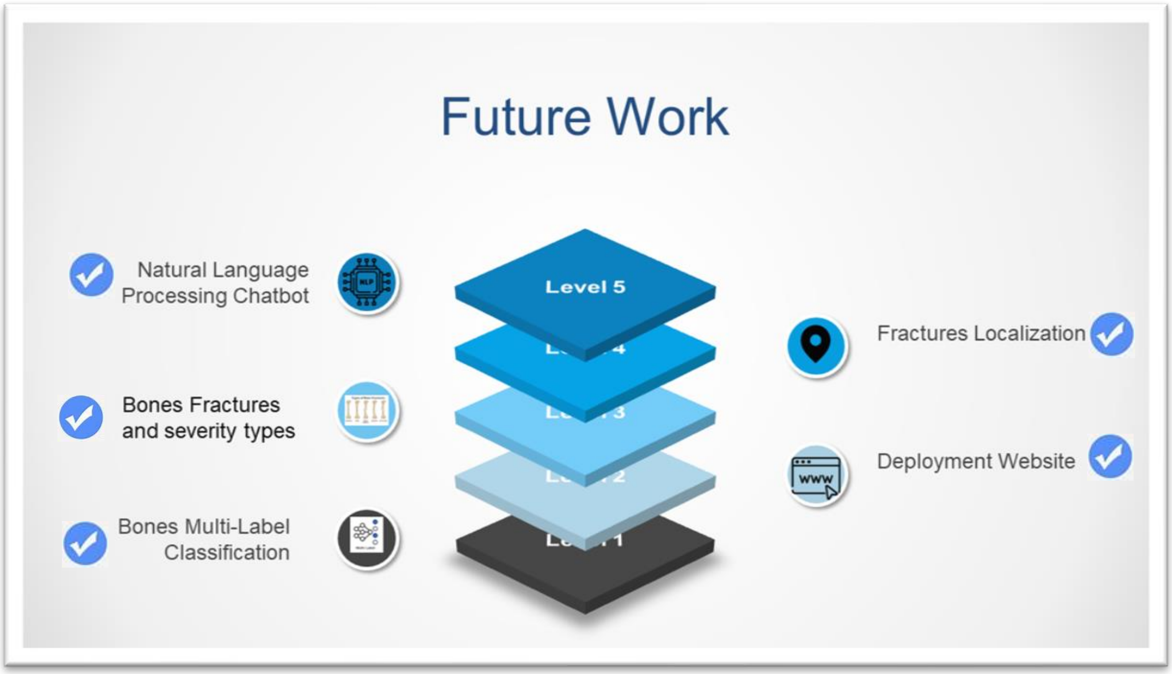


Figure 37 levels of Future work

1-Leve1 (Bones Multi-Label Classification):

We want FRACSCAN to predict multi labels as bone normal or abnormal (fractured) and Bone types, that prevents us to make two models or two SoftMax layers to decrease model’s trainable parameters.

2-Level2 (Deployment Website):

To offer services it’s essential to upload Model and website front end and back end to a good server able to serve users usage successfully, e.g. home screen.

3-Level3 (Bones Fractures and severity types):

Trying to collect data having severity of fracture and its type to cover all details about bone fracture as shown below.

4-Level4 (Fracture Localization):

We have two datasets that have localization annotations to do By Yolo, wrist hand as shown below.

5-Level5 (NLP Chatbot):

Seeking out to find reports, examinations, and cure for patients to collect it as dataset and perform transformers (self-attention context) to generate reports and cure for patients.

References

References of introduction

1- https://www.osteoporosis.foundation/sites/iofbonehealth/files/2019-06/2014_RealMen_FactSheet_Arabic.pdf

2- https://www.osteo-pharma.com/bone-fracture-facts/

3- https://www.thelancet.com/journals/lanhl/article/PIIS2666-7568(21)00172-0/fullt

Reference of related work

4-IBFDS: Intelligent bone fracture detection system - ScienceDirect

5-(PDF) Bone Fracture Detection and Classification using Deep Learning Approach (researchgate.net)

6-(PDF) Bone Fracture Detection in X-ray Images using Convolutional Neural Network (researchgate.net)

7-(PDF) Bone fracture detection using Image Processing (researchgate.net)

8-(PDF) Artificial intelligence application in bone fracture detection (researchgate.net)

9-Bone-Fracture-Detection/.idea/inspectionProfiles at main · manhph2211/Bone-Fracture-Detection (github.com)

References of dataset

[1] https://stanfordmlgroup.github.io/competitions/mura/

[2] https://figshare.com/articles/dataset/GRAZPEDWRI-DX/14825193

[3] https://physionet.org/content/vindr-spinexr/1.0.0/

[4] https://aimi.stanford.edu/lera-lower-extremity-radiographs

[5] https://figshare.com/articles/dataset/The_dataset/22363012

References of methodology

1)inception v3

https://medium.com/@AnasBrital98/inception-v3-cnn-architecture-explained-691cfb7bba08

https://iq.opengenus.org/inception-v3-model-architecture/

https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/

2)efficient net

https://blog.roboflow.com/what-is-efficientnet/

https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142