A PROJECT REPORT ON



DETECTION OF KNEE OSTEOARTHRITIS USING CONVOLUTIONAL NEURAL NETWORKS(CNN)

**Submitted in partial fulfillment of requirements for the award of the degree of**

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING & BUSINESS SYSTEMS**

**Submitted by:**

**P. UDAYA SRI (20091A3449)**

**M. PHANEENDRA BABU (20091A3428)**

**S. RABEEHA (20091A3429)**

**S. AZEEZ BASHA (21095A3401)**

**Under the Guidance of**

**Mr. V. P. Hara Gopal M. Tech, (Ph. D)**

**Assistant Professor, Dept. of CSE&BS**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING & BUSINESS SYSTEMS**

**RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)**

*Approved by AICTE, New Delhi; Affiliated to JNTUA-Ananthapuramu, Accredited by NBA (6-Times); Accredited by NAAC with ‘A+’ Grade (Cycle-3), New Delhi;*

*World Bank Funded Institution; Nandyal (Dist)-518501, A.P*

(Estd-1995)

**YEAR: 2023-2024**

**Rajeev Gandhi Memorial College of Engineering &Technology**

**(AUTONOMOUS)**

*Approved by AICTE, New Delhi; Affiliated to JNTUA-Ananthapuramu, Accredited by NBA (6-Times); Accredited by NAAC with ‘A+’ Grade (Cycle-3), New Delhi;*

*World Bank Funded Institution; Nandyal (Dist)-518501, A.P*

**(ESTD – 1995)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING & BUSINESS SYSTEMS**

**CERTIFICATE**

This is to certify that **P. UDAYA SRI** (*20091A3449*), **M. PHANEENDRA BABU** (*20091A3428*),

**S. RABEEHA** (*20091A3429*) and **S. AZEEZ BASHA** (*21095A3401*) of IV- B. Tech

II- semester, have carried out the major project work entitled “**DETECTION OF KNEE OSTEOARTHRITIS USING CONVOLUTIONAL NEURAL NETWORKS(CNN)**” under

the supervision and guidance of **Mr. V. P. Hara Gopal,** Assistant Professor, CSE&BS Department, in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering & Business systems** from **Rajeev Gandhi Memorial College of Engineering & Technology (Autonomous)**, Nandyal is a bonafied record of the work done by them during 2023-2024.

**Project Guide**



**Mr. V. P. Hara Gopal M.Tech, (Ph.D)**

Assistant Professor, Dept. of CSE&BS

**Place:** Nandyal

**Head of the Department Dr. G. Kishor Kumar M.Tech, Ph.D** Professor, Dept. of CSE&BS

**External Examiner**

***Candidate’s Declaration***

We hereby declare that the work done in this project entitled **“DETECTION OF KNEE OSTEOARTHRITIS USING CONVOLUTIONAL NEURAL NETWORKS(CNN)”**

submitted towards completion of major project in *IV Year II Semester of B. Tech (CSE&BS)* at the **Rajeev Gandhi Memorial College of Engineering & Technology**, Nandyal. It is an authentic record our original work done under the esteemed guidance of **Mr. V. P. Hara Gopal,** Assistant Professor, Department of **Computer Science and Engineering & Business Systems**, RGMCET, Nandyal.

We have not submitted the matter embodied in this report for the award of any other Degree in any other institutions for the academic year 2023-2024.

**By**

**P. Udaya Sri**

**M. Phaneendra Babu**

**S. Rabeeha**

**S. Azeez Basha**

Dept. of CSE&BS, RGMCET.

**Place:** Nandyal

**ACKNOWLEDGEMENT**

We manifest our heartier thankfulness pertaining to your contentment over our project guide **Mr. V. P. Hara Gopal,** Assistant Professor of Computer Science and Engineering & Business Systems department, with whose adroit concomitance the excellence has been exemplified in bringing out this project to work with artistry.

We express our gratitude to **Dr. G. Kishor Kumar garu,** Head of the Department of Computer Science and Engineering & Business Systems department, all the **Teaching Staff Members** of the Computer Science and Engineering & Business Systems department of Rajeev Gandhi memorial College of Engineering and Technology for providing continuous encouragement and cooperation at various steps of our project successful.

Involuntarily, we are perspicuous to divulge our sincere gratefulness to our Principal, **Dr.**

**T. Jaya Chandra Prasad garu,** who has been observed posing valiance in abundance towards our individuality to acknowledge our project work tangentially.

At the outset we thank our honorable **Chairman Dr. M. Santhi Ramudu garu,** for providing us with exceptional faculty and moral support throughout the course.

Finally, we extend our sincere thanks to all the non- teaching **Staff Members** of CSE&BS Department who have co-operated and encouraged us in making our project successful.

Whatever one does, whatever one achieves, the first credit goes to the **Parents** be it not for their love and affection, nothing would have been responsible. We see in every good that happens to us their love and blessings.

**BY**

**P. Udaya Sri** (20091A3449)

**M. Phaneendra Babu** (20091A3428)

**S. Rabeeha** (20091A3429)

**S. Azeez Basha** (21095A3401)

**ABSTRACT**

A frequent kind of arthritis, knee osteoarthritis is characterised by sclerosis, joint space narrowing, osteophyte growth, and bone deformities that can be seen on radiographs. Radiography is the most affordable and widely accessible method, and it is considered to be the best. The Kellgren and Lawrence (KL) grading technique is used to classify X-ray pictures in accordance with the progression of osteoarthritis from normal to severe. Degeneration of osteoarthritis in the knee can be slowed down by early identification, which can aid in early treatment. Regretfully, in an effort to enhance the performance of their models, the majority of currently used methods either combine or eliminate confusing grades. The objective of this research is to present an approach by leveraging an ensemble of CNN models, specifically MobileNet, ResNet, and AlexNet architectures. The choice of using a Convolutional Neural Network (CNN) for knee osteoarthritis classification is driven by its capacity to leverage deep learning techniques for medical image analysis. CNNs excel at feature extraction from medical images, making them ideal for identifying subtle patterns indicative of osteoarthritis. This approach improves the potential to automate diagnosis, reduce human error, and patient outcomes by enabling timely intervention, underscoring its relevance in the realm of medical image analysis. An Osteoarthritis Initiative (OAI) based dataset of knee joint X-ray images is chosen for this study. The dataset was split into the training, testing, and validation set with a 7: 2: 1 ratio. Our results shows that the ensemble approach significantly outperforms individual model predictions, achieving an accuracy of 96%. This improvement underscores the potential of using deep learning ensembles in medical image analysis, offering enhanced diagnostic processes in KOA classification.

Keywords— Knee Osteoarthritis (KOA), Osteoarthritis dataset, CNN, AlexNet, ResNet, Mobile Net, Ensemble

**CONTENTS**

**Chapter No Title Page No**

|  |  |  |
| --- | --- | --- |
| **1.** | **INTRODUCTION**  Project Background |  |
| 1.1 | 2-3 |
| 1.2 | Objectives | 3 |
| **2** | **LITERATURE SURVEY** | 4-5 |
| **3.** | **SYSTEM ANALYSIS**  Existing System |  |
| 3.1 | 6 |
| 3.2 | Proposed System | 6 |
| 3.3 | Architectures | 9 |
| 3.3.1 | MobileNet | 9-10 |
| 3.3.2 | AlexNet | 10-11 |
| 3.3.3 | ResNet | 11-12 |
| **4.** | **FEASIBILITY STUDY** |  |
| 4.1 | Technical Feasibility | 13 |
| 4.2 | Operational Feasibility | 13-14 |
| 4.3 | Economic Feasibility | 14 |
| **5.** | **SYSTEM REQUIREMENT SPECIFICATION** |  |
| 5.1 | Requirement Analysis | 15 |
| 5.2 | Software Requirements Specification (SRS) | 15 |
| 5.3 | Functional Requirements | 15 |
| 5.4 | Non-Functional Requirements | 16 |
| 5.5 | Software Requirements | 16 |

|  |  |  |
| --- | --- | --- |
| 5.6 | Hardware Requirements | 16 |
| 5.7 | Software Installation for this project | 16 |
| **6.** | **MODULES**  System |  |
| 6.1 | 20 |
| 6.1.1 | Store Dataset | 20 |
| 6.1.2 | Model Training | 20 |
| 6.1.3 | Model Predictions | 20 |
| 6.2 | User | 20 |
| 6.2.1 | Load Dataset | 20 |
| 6.2.2 | View Dataset | 20 |
| 6.2.3 | Select Model | 20 |
| 6.2.4 | Evaluation | 20 |
| **7.** | **TECHNICAL DESCRIPTION**  Introduction to Python |  |
| 7.1 | 21 |
| 7.2 | Python concepts | 21-22 |
| 7.3 | History of Python | 22 |
| 7.4 | Python Features | 22-36 |
| **8.** | **SYSTEM DESIGN**  UML |  |
| 8.1 | 37 |
| 8.1.1 | Class Diagram | 37 |
| 8.1.2 | Component Diagram | 37 |
| 8.1.3 | Deployment Diagram | 37 |
| 8.1.4 | Object Diagram | 38 |
| 8.1.5 | Use case Diagram | 38 |
| 8.1.6 | Activity Diagram | 38 |
| 8.1.7 | State Machine Diagram | 38 |
| 8.1.8 | Sequence Diagram | 38 |
| 8.1.9 | Collaboration Diagram | 38 |

|  |  |  |
| --- | --- | --- |
| 8.2 | List of UML Notations | 39-40 |
| 8.3 | UML Diagrams | 41 |
| 8.3.1 | Use case Diagram | 41 |
| 8.3.2 | Class Diagram | 42 |
| 8.3.3 | Sequence Diagram | 43 |
| 8.3.4 | Collaboration Diagram | 44 |
| 8.3.5 | Deployment Diagram | 44 |
| 8.3.6 | Activity Diagram | 45 |
| 8.3.7 | Component Diagram | 46 |
| 8.3.8 | ER Diagram | 46-47 |
| 8.3.9 | DFD Diagram | 47-48 |
| **9.** | **TESTING**  Types of Tests |  |
| 9.1 | 49 |
| 9.1.1 | Unit Testing | 49 |
| 9.1.2 | Integration Testing | 49 |
| 9.1.3 | Functional Testing | 50 |
| 9.2 | System Test | 50 |
| 9.2.1 | White Box Testing | 50 |
| 9.2.2 | Black Box Testing | 51 |
| 9.2.3 | Unit Testing | 51 |
| 9.2.4 | Integration Testing | 52 |
| 9.2.5 | Acceptance Testing | 52 |
| **10.** | **RESULTS AND ANALYSIS** |  |
| 10.1 | Evaluation metrics | 53 |
| 10.2 | Screenshots | 54-62 |
| **11.** | **CONCLUSION** | 63 |
| **12.** | **FUTURE WORK** | 64 |
| **13.** | **REFERENCES** | 65-66 |

**LIST OF ABBREVIATIONS**

KOA : Knee Osteoarthritis

KL : Kellgren and Lawrence

CNN : Convolutional Neural Networks OAI : Osteoarthritis Initiative

ReLu : Rectified linear unit

**LIST OF FIGURES**

[Fig:3.2.1 Flow Chart 19](#_TOC_250027)

[Fig:3.2.2 System Architecture 20](#_TOC_250026)

[Fig:3.3.1 MobileNet Architecture 22](#_TOC_250025)

[Fig:3.3.2 AlexNet Architecture 23](#_TOC_250024)

Fig:3.3.3 ResNet Architecture 24

[Fig:5.7.1 Installing Python 29](#_TOC_250023)

[Fig:5.7.2 Installing PyCharm 30](#_TOC_250022)

[Fig:5.7.3 Setup PyCharm 30](#_TOC_250021)

Fig:5.7.4 Installing Package 31

[Fig:8.3.1 Use case Diagram 53](#_TOC_250020)

[Fig:8.3.2 Class Diagram 54](#_TOC_250019)

[Fig:8.3.3 Sequence Diagram 55](#_TOC_250018)

[Fig:8.3.4 Collaboration Diagram 56](#_TOC_250017)

[Fig:8.3.5 Deployment Diagram 56](#_TOC_250016)

[Fig:8.3.6 Activity Diagram 57](#_TOC_250015)

[Fig:8.3.7 Component Diagram 58](#_TOC_250014)

[Fig:8.3.8 ER Diagram 59](#_TOC_250013)

[Fig:8.3.9 DFD Diagram 60](#_TOC_250012)

Fig:9.2.2.1 Testing Cycle 63

[Fig:10.2.1 Dataset 66](#_TOC_250011)

Fig:10.2.2 KL-Grades different levels 66

Fig:10.2.3 Splitting of data 66

[Fig:10.2.4 Splitting of dataset 67](#_TOC_250010)

[Fig:10.2.5 Model Training Accuracy 68](#_TOC_250009)

[Fig:10.2.6 Model Training Loss 68](#_TOC_250008)

[Fig:10.2.7 Confusion Matrix of the model 69](#_TOC_250007)

Fig:10.2.8 ROC & AUC of the model 69

[Fig:10.2.9 Website Interface 70](#_TOC_250006)

[Fig:10.2.10 Registration page 70](#_TOC_250005)

[Fig:10.2.11 Login page 70](#_TOC_250004)

[Fig:10.2.12 Home page 71](#_TOC_250003)

[Fig:10.2.13 Upload file 71](#_TOC_250002)

[Fig:10.2.14 Choose file 72](#_TOC_250001)

[Fig:10.2.15 Select file 72](#_TOC_250000)

Fig:10.2.16 Normal 73

Fig:10.2.17 Doubtful 73

Fig:10.2.18 Mild 73

Fig:10.2.19 Moderate 73

Fig:10.2.20 Severe 74

**LIST OF TABLES**

Table 8.2: List of UML Notations 51-52

1. **INTRODUCTION**

A frequent kind of arthritis, knee osteoarthritis is characterised by sclerosis, joint space narrowing, osteophyte growth, and bone deformities that can be seen on radiographs. Radiography is the most affordable and widely accessible method, and it is considered to be the best. The Kellgren and Lawrence (KL) grading technique is used to classify X-ray pictures in accordance with the progression of osteoarthritis from normal to severe. Degeneration of osteoarthritis in the knee can be slowed down by early identification, which can aid in early treatment. Regretfully, in an effort to enhance the performance of their models, the majority of currently used methods either combine or eliminate confusing grades. The objective of this research is to present an approach by leveraging an ensemble of CNN models, specifically MobileNet, ResNet, and AlexNet architectures. The choice of using a Convolutional Neural Network (CNN) for knee osteoarthritis classification is driven by its capacity to leverage deep learning techniques for medical image analysis. CNNs excel at feature extraction from medical images, making them ideal for identifying subtle patterns indicative of osteoarthritis. This approach improves the potential to automate diagnosis, reduce human error, and patient outcomes by enabling timely intervention, underscoring its relevance in the realm of medical image analysis. An Osteoarthritis Initiative (OAI) based dataset of knee joint X-ray images is chosen for this study. The dataset was split into the training, testing, and validation set with a 7: 2: 1 ratio. Our results shows that the ensemble approach significantly outperforms individual model predictions, achieving an accuracy of 96%. This improvement underscores the potential of using deep learning ensembles in medical image analysis, offering enhanced diagnostic processes in KOA classification.

**Keywords**— Knee Osteoarthritis (KOA), Osteoarthritis dataset, CNN, AlexNet, ResNet, Mobile Net, Ensemble model

* 1. **Project Background**

Multiple factors contribute to the difficulty of diagnosing, detecting, and treating osteoarthritis (OA). It is a long-term degenerative condition that causes cartilage to deteriorate and finally breaks down bones. One kind of osteoarthritis that affects the knee joint is called knee osteoarthritis (KOA). Pain, stiffness, swelling, and restricted joint movement are examples of physical symptoms. Age, gender, race, genetics, obesity, injuries, low vitamin D levels, and lifestyle choices are risk factors. There are many phases of severity and the disease progresses over time. A recent study found that 16% of people worldwide had KOA. The World Health Organization (WHO) reports that this condition affects adults over 60 globally and is more common in women (18.0%) than in males (9.6%). X-rays, arthroscopy, magnetic resonance imaging [MRI], and symptoms are typically used to diagnose knee osteoarthritis. But OA's early phases are frequently concealed. Furthermore, there is only a limited correlation between the image-represented severity level of OA and the degree of pain and impairment. Therefore, a more accurate diagnostic method is required to identify OA in its early phases. In this case, OA-related biomarkers may be useful.

The basis for identifying and treating KOA is the use of radiographs or X-rays to evaluate restlessness and pain. Cyst formation, subchondral sclerosis, osteophytes, and joint space narrowing (JSN) are important characteristics that can be seen on X-rays. The lack of protective cartilage between knee joints is referred to as JSN. A bony bump developed on bones or joints is called an osteophyte, and abnormal increase in bone density is called subchondral sclerosis.

We have made an effort to close this gap in this study by raising the prediction accuracies for every KL grade. A ten-year prospective observational study of KOA involving multiple centers is called the Osteoarthritis Initiative (OAI). With funding from the National Institutes of Health (a division of the Department of Health and Human Services), they enrolled 4796 men and women. The data they used came from over 431,000 imaging and clinical visits, and there were about 26,626,000 images in this archive. Our study's knee X-ray pictures were taken from this dataset.

Our goal was to increase the model's performance across all grades while concentrating on early KOA detection. Physicians can develop a more effective early KOA treatment plan with the aid of early diagnosis and accurate KOA grade prediction, which will also save patient costs associated with delayed discovery.

* 1. **Objectives**

1. Assess how well deep learning models categorize the degree of severity of knee injuries.
2. Evaluate the ResNet, MobileNet, and AlexNet architectures' effectiveness.
3. Examine how deep learning affects knee injury diagnosis based on X-rays.
4. Recognize the advantages, disadvantages, and potential paths forward for knee injury detection.
5. Address issues such as interpretability and data imbalance in order to improve diagnosis.
6. **LITERATURE SURVEY**

**[1]. Y. Badshah, M. Shabbir, H. Hayat, Z. Fatima, A. Burki, S. Khan, and S. U. Rehman, ‘‘Genetic markers of osteoarthritis: Early diagnosis in susceptible Pakistani population,’’ J. Orthopaedic Surgery Res., vol. 16, no. 1, pp. 1–8, Dec. 2021.**

Knee Osteoarthritis is a degenerative disease that affects many individuals in their forties or later and can lead towards impairment of physical functions and poor quality of life. Our study aimed to compare the effectiveness of dry needling with conventional physical therapy in patients with knee osteoarthritis (OA). It was a non-randomized clinical study conducted after ethical approval at Syed Medical Complex and Amin Welfare and Teaching Hospital Sialkot in 6 months. Sample size was calculated to be 58 using WHO calculator. Purposive sampling technique was used to enrol the participants according to predetermined eligibility criteria. They were allocated into two groups. Group A received conventional physical therapy treatment and group B received dry needling for 3 weeks, 2 sessions per week. Western Ontario and McMaster Universities Osteoarthritis Index (WOMAC) scale and Numeric Pain Rating Scale (NPRS) were used to measure the outcomes before 1st treatment session and after last session. Paired sample *t*-test and independent sample *t*-test were applied to compare the results, and *p*≤0.05 was considered significant. Pre- treatment WOMAC score in group A was 50.07±11.835 and in group B it was 45.87±12.512. Post-treatment WOMAC score in group A was 38.87±13.731 and in group B it was 24.33±8.926. The scores improved in both groups, but group B showed significant improvement.

**Conclusion:** Both conventional physical therapy and dry needling are effective to manage pain and functional limitation in patients with knee OA. Dry needling proved more effective compare to conventional physical therapy.

**[2]. S. K. Das and A. Farooqi, ‘‘Osteoarthritis,’’ Best Pract. Res. Clin. Rheumatol., vol. 22, no. 4, pp. 657–675, 2008.**

Historically disease knowledge development and treatment innovation in osteoarthritis (OA) has been considered to be slow. One of the many reasons purported as responsible for this slow pace has been the alleged lack of valid and responsive biomarkers to ascertain efficacy, which itself has been dependent upon the slow evolution of the understanding of the complex nature of joint tissue biology. This narrative review outlines the rationale for why we need OA biomarkers with regard to biomarker validation and qualification. The main biomarkers in current development for OA are biochemical and imaging markers. We describe an approach to biomarker validation and qualification for OA clinical trials that has recently commenced with the Foundation of NIH OA Biomarkers Consortium study cosponsored by the Osteoarthritis Research Society International (OARSI). With this approach we endeavor to identify, develop, and qualify biological

markers (biomarkers) to support new drug development, preventive medicine, and medical diagnostics for osteoarthritis.

**[3]. S. S. Gornale, P. U. Patravali, and R. R. Manza, ‘‘Detection of osteoarthritis using knee X-ray image analyses: A machine vision based approach,’’ Int. J. Comput. Appl., vol. 145, no. 1, pp. 20–26, 2016.**

Significant information extraction from the images that are geometrically distorted or transformed is mainstream procedure in image processing. It becomes difficult to retrieve the relevant region when the images get distorted by some geometric deformation. Hu's moments are helpful in extracting information from such distorted images due to their unique invariance property. This work focuses on early detection and gradation of Knee Osteoarthritis utilizing Hu's invariant moments to understand the geometric transformation of the cartilage region in Knee X-ray images. The seven invariant moments are computed for the rotated version of the test image. The results demonstrated are found to be more competitive and promising, which are validated by ortho surgeons and rheumatologists.

**[4]. M. N. Iqbal, F. R. Haidri, B. Motiani, and A. Mannan, ‘‘Frequency of factors associated with knee osteoarthritis,’’ J. Pakistan Med. Assoc., vol. 61, no. 8, p. 786, 2011**

A cross sectional study was conducted in the department of medicine at Liaquat National Hospital Karachi from September 2007 - March 2008 on patients diagnosed with osteoarthritis of knee. One hundred patients of age more than 18 years, and of either gender were consecutively included. Those patients with family history, occupations like farmers, mill workers and jack hammer operators, females with hormonal replacement therapy, patients with hyperparathyroidism, haemochromatosis and systemic lupus erythematosus were excluded. The patients were then graded on the basis of knee joint X-ray findings into four grades. The main outcome variables of the study were factors associated with osteoarthritis which included obesity, age, gender, smoking and anemia.

1. **SYSTEM ANALYSIS**
   1. **Existing system**

The current system for detecting knee osteoarthritis employs a Convolutional Neural Network (CNN) with MobileNet architecture for classification. Although it can analyze medical imaging data to identify and classify the severity of the disease, it has several drawbacks that can result in inaccurate results and limited generalization. These include data dependence, high resource requirements, interpretability issues, data privacy and security concerns, and false positives/negatives.

**Disadvantages of Existing System**

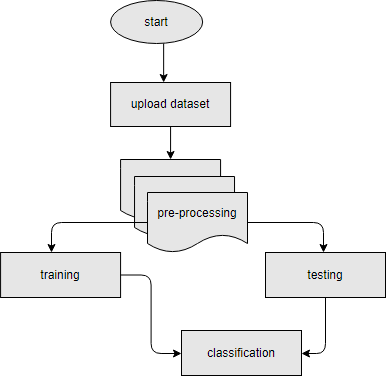
1. Model size and less computational efficiency.
2. Low accuracy.
3. Limited resource constraints.
   1. **Proposed system**

Our new method aims to improve the detection of knee osteoarthritis by using a specialized Convolutional Neural Network (CNN) combined with AlexNet, MobileNet, and ResNet architectures. We start by carefully processing the data to optimize it for training. Then, we train and evaluate the CNN model enriched with features from AlexNet, MobileNet, and ResNet. We use key metrics to guide optimization and gain insights into the strengths and limitations of each architecture. This multi-architectural approach aims to improve the accuracy of knee osteoarthritis detection, showcasing the potential advancements achievable through the fusion of CNN, AlexNet, MobileNet, and ResNet capabilities in medical diagnostics.

**Advantages of Proposed System**

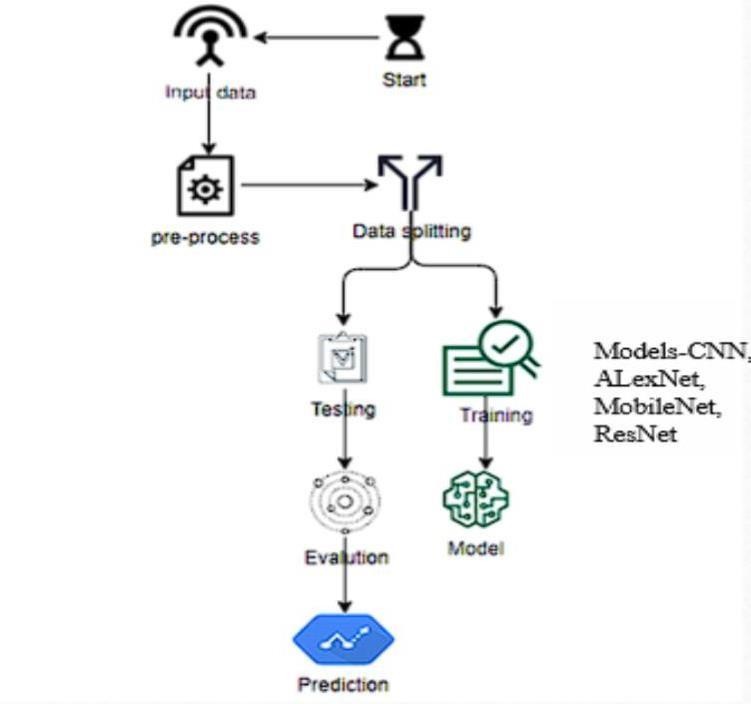
1. Improved accuracy.
2. High efficiency.
3. Increased computational speed.

**Flow of the project**



# Fig:3.2.1 Flow Chart

**System Architecture**



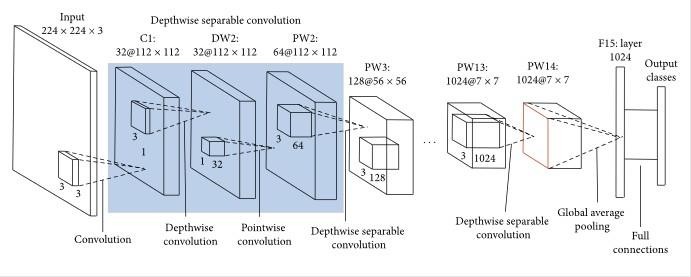
# Fig:3.2.2 System Architecture

* 1. **Architectures**
     1. **MobileNet**

Our knee osteoarthritis detection method is built on top of MobileNet, a convolutional neural network optimized for performance on mobile and edge devices. This network, which was created in 2017 by Howard et al., puts computational efficiency first without sacrificing performance. Using depth wise separable convolutions, MobileNet achieves a lightweight design with a depth of 28 layers. Because of this innovation, there are a lot less factors in the model, which makes it ideal for use in contexts with limited resources.

By fine-tuning, MobileNet- which was trained via transfer learning on a carefully selected dataset of knee X-ray ages adapts to different degrees of osteoarthritis severity. Standard criteria like accuracy, precision, recall and F1 score are used to assess the model's performance, giving rise to a thorough evaluation of its efficacy. Particularly for real-time processing of knee X-ray images in clinical settings with restricted computational resources, MobileNet's small size adds to its appeal. Due to its high computing efficiency, MobileNet can be used to improve knee-related pathology diagnosis processes. To summarise, our system for detecting osteoarthritis in the knee has been enhanced by using MobileNet, which shows how lightweight architectures may be utilised to reliably and efficiently analyse medical images, hence improving diagnostic capabilities.

# Fig:3.3.1 MobileNet Architecture



* + 1. **AlexNet**

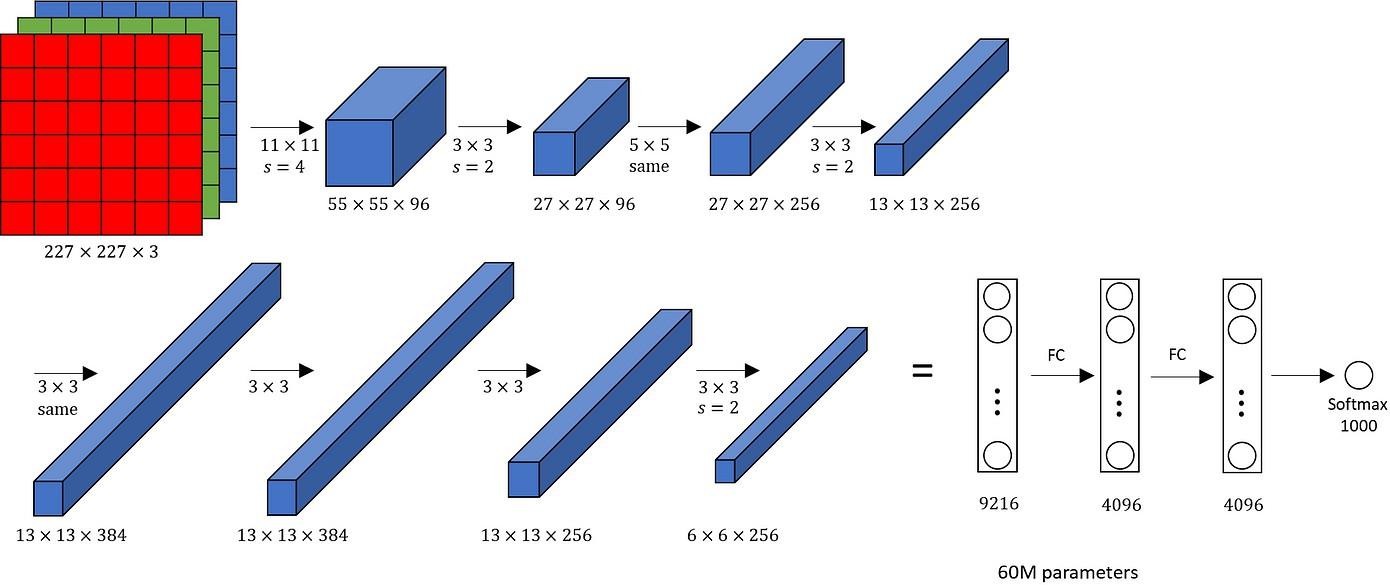
The accuracy and effectiveness of our knee osteoarthritis diagnosis system have been improved with the addition of AlexNet, a cutting-edge convolutional neural network. The ImageNet Large Scale Visual Recognition Challenge was where AlexNet rose to popularity, which was developed in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton.

Rectified linear units (ReLU) and local response normalisation are two of AlexNet's architectural features that set it apart from other networks with a substantial depth of eight layers. Complex features that are essential for image classification applications can be extracted thanks to this deep architecture.

The AlexNet-enhanced model is trained on a variety of knee X-ray images and then goes through transfer learning and fine-tuning to adjust to the subtleties of osteoarthritis identification. The dataset's wide range of severity levels guarantees that the model can correctly classify diverse knee diseases. Standard metrics including accuracy, precision, r ecall, and F1 score are used in performance evaluation to give a thorough grasp of the model's diagnostic capabilities. Because of its capacity to classify images and its flexibility when it comes to medical imaging, AlexNet can be a useful tool for diagnosing osteoarthritis in the knee.

Despite not being as light as other more recent architectures, AlexNet's strong features and track

record of performance make it important for medical image analysis. AlexNet's ability to improve diagnostic precision and progress the field of knee-related pathology identification is demonstrated by its inclusion into our system.



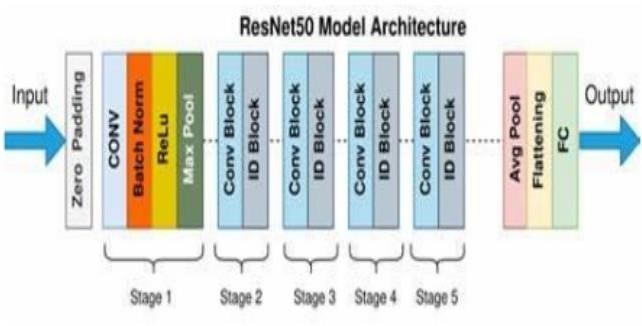
# Fig:3.3.2 AlexNet Architecture

* + 1. **ResNet**

ResNet50 is a deep convolutional neural network (CNN) architecture renowned for its pioneering introduction of residual connections, which revolutionized the training of extremely deep networks. Comprising a total of 50 layers, ResNet50 design includes residual blocks that incorporate skip connections. These skip connections enable gradients to propagate more efficiently during training by circumventing the vanishing gradient problem, which often hampers the training of deep networks.

A distinctive feature of ResNet50 is its utilization of bottleneck blocks, strategically structured to enhance computational efficiency. These bottleneck blocks begin with 1x1 convolutions that reduce the dimensionality of the feature maps, followed by 3x3 convolutions to capture and process spatial information. This design choice not only reduces computational costs but also facilitates the training of deeper networks by effectively managing the flow of information through the network layers.

By incorporating residual connections and bottleneck blocks, ResNet50 achieves remarkable performance in various computer vision tasks, surpassing previous architectures in terms of accuracy and scalability. Its ability to mitigate the challenges associated with training deep networks has made ResNet50 a cornerstone in the field of deep learning, inspiring subsequent advancements in CNN architecture design and training methodologies.



**Fig:3.3.3 ResNet**

1. **FEASIBILITY STUDY**

A key phase in the software development process is the feasibility study. The viability of the project and the likelihood that the system will benefit the company are both examined in the preliminary investigation. Examining the technical, operational, and financial viability of adding new modules and fixing outdated systems is the primary goal of the feasibility study.

A feasibility study ought to be carried out using a range of standards and guidelines.

There are three different types of feasibility studies: technical, operation, and economic.

* 1. **Technical Feasibility**

During the feasibility phase of the research, the following technical problems are usually brought up:

* Is the technology required to carry out the specified actions available?
* Can the suggested equipment technically store the data needed to operate the new system?
* Will the suggested system be able to adequately respond to questions from users, irrespective of their quantity or location?
* Should the system be created, is it upgradeable?
  1. **Operational Feasibility**
* **User-friendly:** Customers will utilize the forms for a variety of tasks, such as reviewing the information of existing routes and adding new ones. Additionally, the customer requests that the reports display the different transactions according to the limitations. These reports and forms are produced with the client in mind.
* **Dependability:** The package will retrieve ongoing online transactions. The user will input the previous transactions into the system.
* **Security:** Hacking, viruses, and other threats should be prevented on the web server and database server.
* **Portability:** Standard open-source software such as Java, the Tomcat web server, the Internet Explorer browser, and so forth will be used in the development of the program

(apart from Oracle). These programs can be used with Linux and Windows operating systems. Therefore, issues with portability won't arise. Availability: There will alwaysbe access to this program.

* **Maintainability:** The 2-tier design is used by the "wheels" system. The GUI, also referred to as the front-end, is the first tier, and the database, which runs on My-Sql and is the second tier, is where the back-end components don't exist
  1. **Economic Feasibility**

Even if the system is technically possible and will be used, it will still be a smart investment for the company. In economic feasibility analysis, process improvement costs are measured against the benefits of the new process. Financial benefits must equal or exceed expenses. This system can be profitable

1. **SYSTEM REQUIREMENT SPECIFICATION**
   1. **Requirement analysis**

Requirements Analysis is a software engineering project that bridges the gap between system- level software deployment and software design. Requirements analysis shows the software function and the function of presenting the software in relation to other elements. These user-required features can be formal or informal. In the legal system, the user clearly defines the purpose of the software. This provides a good basis for software engineers to conduct requirements analysis. Informally, the purpose and output are not clearly defined by the user. Software engineers are responsible for achieving goals and results by engaging more users.

* 1. **Software Requirements Specification (SRS)**

Software requirements specification A requirements specification for a software system is a complete description of the behavior of the system to be designed and may include a set of use cases that describe the interaction between the user and the software. It also has non-functional requirements. Non-functional regulations impose restrictions on design or use. We need a clear and precise understanding of the product to be created to meet the needs. This is planned after detailed communication with the team and customers. An effective SRS outlines how an application will communicate with the hardware of the system, with other applications, and with human users in a wide range of real-world scenarios.

* 1. **Functional Requirements User interface**

The application provides keyboard shortcuts to perform the desired action, usually using the key. They act as shortcuts and provide easy navigation through their software. Use exceptions to complete necessary errors to isolate negative consequences or events. These messages appear to the user as dynamic html or notifications.

* 1. **Non-Functional Requirements**

**Performance requirements**

Good bandwidth, low network congestion. Determine the shortest path to the goal to improve performance. This product is protected against unauthorized users and only authenticated users are allowed to use the application. The users of the system are administrators, users and students.

**Safety requirements:**

No harm is expected from the use of the application either to the OS or any data that resides on the client system.

* 1. **Software Requirements**
     + Software’s : **Python 3.6 or high version**
     + IDE : **PyCharm**
     + Framework : **Flask**
  2. **Hardware requirements**
     + Operating system : **Windows 10**
     + RAM : **8 GB**
     + Hard disc or SSD : **128MB**
     + Processor : **Intel 3rd generation or high or Ryzen with 8 GB Ram**
  3. **Software Installation for this project**

**Installing Python**

1. To download and install Python visit the official website of Python https:/[/www.python.org/downloads/](http://www.python.org/downloads/) and choose your version.

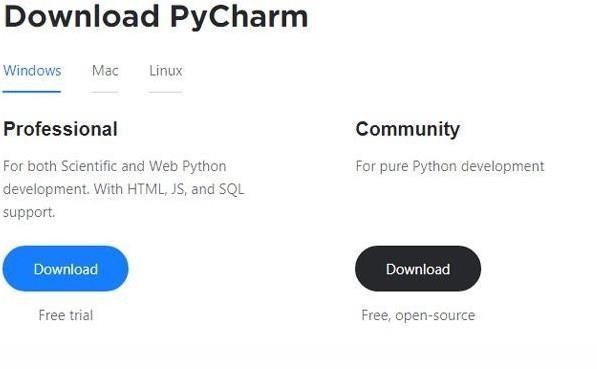


# Fig:5.7.1 Installing Python

1. Launch the Python installation exe when the download has finished. Click Install Now to continue.
2. Python is now installing, as you can see.
3. After it's done, a screen stating that the Setup was successful will appear. Click "Close" now.

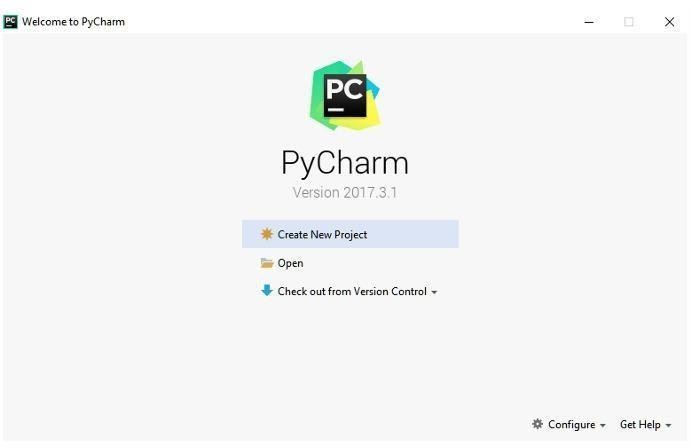
**Installing PyCharm**

1. Go to the website https:/[/www.jetbr](http://www.jetbrains.com/pycharm/download/)a[ins.com/pycharm/download/](http://www.jetbrains.com/pycharm/download/) and select the "DOWNLOAD" option in the Community Area to download PyCharm.
2. When the download is finished, launch the installer for PyCharm. The setup wizard ought to have begun. Choose "Next."
3. Modify the installation path as necessary on the next screen. Choose "Next."
4. Choose "Next" on the following box, where you can if you'd like create a desktop shortcut.
5. Choose a folder from the start menu. Keep JetBrains selected and press "Install."
6. Attend to the installation's completion.



# Fig:5.7.2 Installing PyCharm

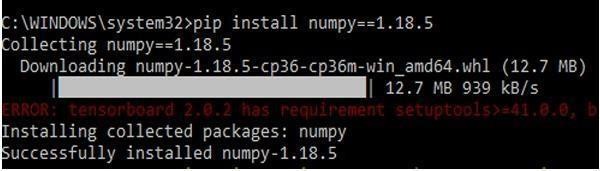
1. Once the installation is complete, you should get a notification indicating that PyCharm has been installed. Click the "Run PyCharm Community Edition" box first, then click "Complete," if you wish to proceed with running it.
2. The following page will show once you click "Complete."



# Fig:5.7.3 Setup PyCharm

1. To carry out your project properly, you must install a few packages.
2. Log in as an administrator to the terminal, anaconda prompt, or command prompt.
3. When the window appears, type "pip install package name" to install the desired package using the supplied path (like NumPy, pandas, sea born, scikit-learn, Matplotlib, Pyplot)

Ex: pip install Numpy



**Fig:5.7.4 Installing Packages**

1. **MODULES**

The Two modules are:

* 1. **System**
     1. **Store Dataset**

The user-provided dataset is stored by the system.

* + 1. **Model Training**

The system feeds the chosen model with the data it has collected from the user.

* + 1. **Model Predictions**

The system uses the information provided by the user to forecast the results.

* 1. **User**
     1. **Load Dataset**

The user can load the dataset he/she wants to work on.

* + 1. **View Dataset**

The User can view the dataset.

* + 1. **Select model**

User can apply the model to the dataset for accuracy.

* + 1. **Evaluation**

User can evaluate the model performance.

1. **TECHNICAL DESCRIPTION**
   1. **Introduction to Python Python**

**What is a script?**

I have concentrated on Python's ability to be utilized for interactive programming. This is an extremely useful feature that lets you launch a program and have interactive elements executed right away.

**What separates a program from a script? Script**

Scripts are usually created by end users, or at least modified by them, and are distinct from the primary program code, which is often written in a different language. Scripts are commonly interpreted from source code or bytes, unlike the programs they oversee, which are often written to native machine code.

**Program**

The computer can use the executable version of the software to start running the instructions right away. The same program written in human-readable source code, from which executable applications are produced.

* 1. **Python concepts**

If you don't want to learn about the ins and outs of Python, feel free to go on to the next chapter. I hope to show the reader in this chapter why Python is, in my view, one of the best programming languages available and an excellent place to start.

* an open-source general-purpose language.
* Object-oriented, procedural, and functional
* Outstanding interactive setting;
* Simple integration with C/ObjC/Java/Fortran;

Python is an interpreted, object-oriented, interactive, and potent programming language. Python is

designed to be extremely readable. Compared to other languages, it has fewer syntactical structures and frequently uses English terminology instead of punctuation.

* **Python is Interpreted:** Python is handled at runtime by the parser. It is not necessary to assemble your software before launching it. This is similar to how PHP and PERL work.
* **Python is Interactive:** You can really sit at a Python prompt and start a conversation with the interpreter right away while building a program.
* **Python is Object-Oriented:** Python is an object-oriented language because it supports object- oriented programming, which turns code into objects.
* **Python is an introductory language:** Python is an excellent language for new programmers and makes it easier to create a wide range of applications.
  1. **History of Python**
* In the late 1980s and early 1990s, Guido van Rossum invented Python in the Netherlands' National Research Institute for Mathematics and Computer Science.
* Python has direct ancestors in ABC, Modula-3, C, C++, Algol-68, Smalltalk, UNIX shell, and other scripting languages.
* Python has been safeguarded by copyright. Similar to Perl (GPL), Python source code is now available under the GNU General Public License.
* Even though Python is now maintained by the institution's core development team, Guido van Rossum still has an important role in shaping the language's future.
  1. **Python Features**
* **Simple and readable syntax:** Python's syntax is clean, easy to read, and emphasizes readability, making it ideal for both beginners and experienced programmers.
* **Interpreted and interactive:** Python is an interpreted language that enables immediate feedback and interactive development tools such as the Python interpreter or Jupyter notebooks.
* **High-level language:** Python abstracts from low-level details, allowing programmers to focus on solving problems rather than managing memory or hardware details.
* **Extensive standard library:** Modules make up the large standard library of Python. Reduces the need for external dependencies for a variety of tasks such as file I/O, networking, data manipulation, and network development.
* **Dynamic typing:** Python is dynamically typed, which means that variable types are determined at runtime, providing flexibility and ease of use.
* **Object-Oriented Programming:** Python supports Object-Oriented Programming (OOP) paradigms, which allow you to create reusable and modular code using classes and objects.
* **Functional Programming:** Python also supports functional programming concepts, including first-class functions, lambda expressions, and higher-order functions
* **Pure cross-platform compatibility:** Python code can run on multiple operating systems, including Windows, macOS, and Linux, making it highly portable.
* **Large community and ecosystem:** Python has a vibrant and active community. developers who contribute to an ecosystem of libraries, frameworks and tools for many different industries, such as data science, machine learning, web development and more.
* **Easy integration:** Python can be easily integrated with other programming languages and technologies, allowing seamless interoperability and integration with existing systems.

A few of the many helpful features that Python has to provide, in addition to the ones mentioned above, are listed below.

* Functional and structured programming techniques are supported, as well as OOP.
* It can be used as a scripting language that can be converted into byte-code or to develop large applications.
* It provides extremely high-level dynamic data types and permits dynamic type checking.
* It makes automated trash collection possible.
* Integration with Java, C, C++, COM, ActiveX, and CORBA can be completed quickly and easily.

**Variables**

Variables are just predefined memory locations where values can be kept. This suggests that when you create a variable, you allocate some memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be put in the reserved memory. Because of this, you can give these variables other data types so that they can store characters, decimals, or integers.

**Standard Data Types**

Memory is able to store a variety of data types. For example, an individual's age is recorded as a number, yet their address is stored as a string of alphanumeric characters. Python contains a variety of standard data types that explain the various operations that may be performed on them as well as the corresponding storage mechanism.

There are five common data types in Python.

* Dictionary
* Numbers
* String
* List
* Tuple

**Python Numbers**

Number data types are used to hold numeric values. When you assign a value to a number object, you create it.

**Python Strings**

A string in Python can be defined as an uninterrupted sequence of characters enclosed by quote marks. Python allows quotations to be used in pairs, single or double.

**Python Lists**

Lists are the most versatile composite data type in Python. Square brackets are used for separating items in lists, while commas ([]) are used to separate them. In C, lists and arrays are similar in certain ways. The fact that each entry in a list may belong to a different data type is one way that they differ from one another.

The values retained in a list can be retrieved using the slice operator ([] and [:]), with indices starting at 0 at the list's beginning and going all the way to -1 at the list's end. The asterisk (\*) denotes the repetition operator, while the plus (+) symbol denotes the list concatenation operator.

**Python Tuples**

A tuple is a sequence data type that is similar to a list. Commas are used to separate the several values that make up a tuple. Unlike lists, though, tuples have parenthesis all around them.

Tuples and lists differ primarily in that tuples are enclosed in parenthesis (()) and are immutable, whereas lists are enclosed in brackets ([]) and allow for item and size modifications. The same is true for tuples and read-only lists.

**Python Dictionary**

Python makes use of dictionaries, which resemble hash tables. They function similarly to associative arrays or hashes found in Perl and are composed of key-value pairs. Dictionary keys are often texts or numbers, although they can be nearly any sort of Python data.Conversely, values can be any arbitrary Python object.

Square brackets are used to assign and access values, while curly braces are used to surround dictionaries ([]).

**Different modes in python**

The two basic modes of Python are normal and interactive respectively.

The finished and scripted.Py files are executed in the standard way by the Python interpreter.

When in interactive mode, a command line shell runs previously issued commands in active memory and gives immediate response after each statement. As new lines are fed into the interpreter, the feed program is reviewed in its entirety.

A few Python libraries are:

* Random
* NumPy
* Pymysql
* Pandas.

**Pandas**

* Pandas provide us access to a number of Data Frames and Series. It makes basic data exploration, organizing, visualization, and manipulation possible.
* Pandas' intelligent indexing and alignment features provide you the best possible data labeling and organization.
* The code in this package is so clean that even those with no programming knowledge can easily work with it.
* Pandas has certain unique features that let you handle missing values or data with the appropriate measure.
* You can read and write data in a variety of databases, data formats, and web services using its integrated tool set.
* Pandas supports a number of different formats in addition to JSON, Excel, CSV, and HDF5. Actually, Pandas lets you link multiple databases at once.

**NumPy**

* NumPy's arrays offer modern mathematical computations on massive volumes of data. Because of NumPy, these activities may be completed much more easily and simply.

NumPy has added masked arrays to its list of array objects. It also has a lot of other characteristics, such as the capacity to work with logical structures, carry out general linear algebra and discrete Fourier transforms. Whenever you change an N-dimensional array's form.

* NumPy will create new arrays and remove the old ones. This Python package has useful integration tools.
* NumPy is easily included using computer languages such as C, C++, and FORTRAN code. NumPy provides capability similar to that of MATLAB.

**Pymysql**

* Pymysql is a Python library that facilitates communication between Python programs and MySQL servers.
* Obtaining the port settings using Python properties.
* The MySQL-Python driver was first poorly ported to become Pymysql, a MySQL driver developed completely in Python.
* Pymysql fulfills all requirements for a driver.
* It is actively maintained, hosted on GitHub, accessible via Pipy, and fully open source.
* It is fully compatible with Python 3 and the event let-monkey patch because it is entirely written in Python.

**Random**

* + You can use it to do random operations, such as creating a random integer, selecting an item at random from a list, randomly rearranging things, etc.
  + Generate random numbers using various distributions, including float and integer distributions.
  + Choosing constituents at random from the population.
  + The attributes of the random module.
  + Confound the sequence data. Random number generator seeding.
  + Generate random strings and passwords.

**Python objects and classes**

They serve as OOP's cornerstone. The class creates a new item. This item could be any kind of object, like an abstract data concept or a chair model. Every class contains unique characteristics that are unique to it, like variables and methods. Classes are "the big thing" and quite strong in most programming languages these days. As a result, the book has multiple chapters later on that are solely focused on OOP.

The class is the most essential component of object-oriented programming. You already know how to use functions to make your application do anything.

It's almost time to step inside the big, scary world of Object-Oriented Programming (OOP). To be very honest, it took me several months to figure out what was going on.Since I knew how functions worked, I did well when I first learned C and C++.

I had dabbled with BASIC in the early 1990s and knew that functions were comparable to subroutines, so there wasn't much new to learn.

However, as my C++ course started talking about objects, classes, and all the other facets of OOP, my scores obviously declined.

Once you know how to utilize it, OOP becomes a pretty powerful tool. Additionally, a lot of Python libraries and APIs make use of classes, so you should at least be aware of what the code does.Keep in mind that your code uses objects while using Python and OOP.Using the most effective method is not required. For instance, you might not require a fully functional class with methods and entire language code if all you want to do is return a count. With Python, you can go as technical as you like. Python has demonstrated in the past that it can operate effectively with functions. You can combine functions and classes as much as you like within the same application, which is not possible with other programming languages like Java. You can now write code as a result.

An object is a single entity that consists of variables and functions. Classes provide objects access to variables and functions. Classes serve as your objects' templates in a way.Here are a few brief OOP tips for Python.• An object of the designated class is created and given a name by the class declaration. Thus, a new namespace is created. Assignments give class properties in a course. The name must be supplied using the dot syntax, ClassName.Attribute, in order to use these attributes. Using class attributes, the object's state and related behavior are exported.

A class's instances are all endowed with these attributes.

* Similar to calling a function, invoking a class also creates a new instance of the class.This is when the bit about multiple copies come in.
* Every instance receives its own namespace and inherits the default class's properties (see "inherit"). As a result, instance objects won't overlap and get cluttered within the program.
* By designating a particular instance, the term itself makes instance-specific features possible. This enables you to customize an instance by adding elements like variables.

**Inheritance**

To begin with, classes provide you the opportunity to modify a program without actually changing it.To explain, instead of changing the existing components of the program by subclassing a class, you can change its behavior by adding new ones.

As we have seen, instances of a class inherit its properties.

But classes can also inherit characteristics from other classes. Because of this, a subclass descended from a superclass, allowing you to construct a general superclass that subclasses can customize. Subclasses can override a superclass's logic, allowing you to modify the behavior of your classes without changing the superclass itself.

**Overloads of Operators**

Operator overloading essentially means that objects that you construct from classes can use the actions (operations) that are already defined inside Python, like addition, slicing, printing, etc.

Overloading is easier to understand and use because it more directly aligns behavior to Python's object model and object interfaces to built-in Python objects, even when class methods can be used to achieve identical tasks. Custom classes can override almost all of Python's built-in operation methods.

**Exceptions**

Although I've covered exceptions previously, I'll go into further detail today. Essentially, exceptions are activities that, whether intentionally or inadvertently, change the flow of a program.

These are certain events that can occur when something goes wrong, like trying to read a file that doesn't exist, or when the program reaches a milestone, like the conclusion of a loop.

Since exceptions are by definition rare occurrences, they belong in their own class and are the "exception to the rule."

The examples below are just a few:

* If a dictionary key that doesn't exist is accessed, a Key Error exception will be raised.
* If a value is searched for in a list and it is not present, a Value Error exception will be raised.
* Calling a non-existent method will result in the throwing of an Attribute Error exception.
* If a variable that doesn't exist is referenced, a Name Error exception will be raised.
* If data types are merged without authorization, a Type Error exception will be raised.
* Previously, we have seen how exceptions can be used to catch failures and keep programs running while talking about files.

This is the way that exceptions are most commonly used.

Frequently, your software is so short that an anomaly won't make a big difference.

It is also possible to swiftly and readily ascertain whether your code logic is wrong if the mistake happens at the command line.

Nevertheless, your actual software would crash and cease to function if the identical error occurred in it. It is possible to manually add exceptions to the code by raising an exception.

It works exactly like a system-caused exception, but instead of happening by accident, the programmer is doing it on purpose. This could have a number of reasons. The benefit of using exceptions is that they don't introduce any processing overhead into the code by definition.Because exceptions aren't expected to arise often, they aren't handled until they do.

You can think of the if/elif statements as a particular type of exception. In practical terms, blocks and exceptions both let you accomplish the same thing. However, as was previously said, whereas exceptions are only handled when they occur, blocks are processed continuously. When exceptions are used appropriately, your software will perform better.

When failures are more likely to happen, it is preferable to use exceptions; if blocks make Python always check for more conditions before continuing.

If you incorporate error-handling statements into your programming logic, it may be difficult to understand, modify, and debug your software. Additionally, exceptions simplify code maintenance.

Specific Exemptions are :

It is possible to create custom exceptions in Python, but I won't go into too much detail about that here.It's normally not necessary to do this often, but having the option is still a good thing.Before making your own exception, be sure there aren't any already-existing ones that will work for you.

Having been "tried by fire" over the years, they have not only proven to be reliable and bug-free but also performance-optimized.The next chapter will cover object-oriented programming, which involves making your own exceptions.

When constructing a custom exception, the programmer choose which base exception to use as the class to inherit from; for instance, the Arithmetic Error exception class would likely be used if an exception was created for imaginary or negative integers.

**Python Modules**

Python files, also known as modules, are where we can store our code. This is really useful for more serious programming when we don't want to start over on a long function description just to remedy one mistake. In essence, we are building our own collection of modules akin to those in the Python library by doing this. To make this possible, Python has a capability that lets definitions be kept in a file and used in a script or an interactive interpreter instance. Modules are the definitions from a file like this, and they can be imported into other modules or the main module.

**Testing Code**

* As was already said, an editor is usually used to write code in a file.
* To test the code, import it into a Python session and try running it.
* Usually there's a mistake, so you go back to the file, make the necessary corrections, and rerun the test.
* This has to be done repeatedly until you are satisfied that the code works.
* The process as a whole is called the development cycle.
* There are two types of errors that you will come into. Syntax errors occur when the form of a command is not accurate.
* This could happen, among other things, if you type erratically and misspell terms or refer to something by the incorrect name. In Python, a syntax error will always produce an error.

**Functions in Python**

It's quite useful to be able to define our own functions in Python. In general, if you just need to perform a calculation once, use the interpreter. However, create a function if you or others regularly need to perform a particular kind of calculation.

Functions are used in programming to put together a collection of instructions that you want to use frequently or that, because of their complexity, are better left alone in a sub-program and called only when necessary. Thus, a function is a segment of code designed to accomplish a specific task.

The function's input requirements may vary depending on the task at hand. Following completion of the task, the function may or may not return one or more values.

There are three types of functions in Python:

* help()
* min()
* print()

**Python Namespace**

Generally speaking, a namespace—also called a context—is a naming scheme used to distinguish names in order to avoid ambiguity. The initial name and the surname, which is the family name naming convention, are common to everybody.

Networks provide an example. Every network device, including workstations, servers, printers, and so on, needs its own unique identity and address. Another example is the way file systems are organized into directories.

Even though the same file name may be used in many directories, the pathnames provide for a unique retrieval of the files. In computer languages, identifiers are frequently employed in namespaces or contexts. Declared identifiers of a namespace are associated with that namespace.

As a result, several namespaces can specify the same identifier independently.

(For instance, the same file name in many directories) Programming languages that allow namespaces to determine which namespace an identifier belongs to may have a number of criteria that apply.

There is a mapping from names (keys) to objects (values) when namespaces are implemented using Python dictionaries. The user does not need to know this in order to use namespaces and write Python programs.

Python namespaces consist of:

* module global names.
* Local names used when a function or method is called.
* Built-in names: This namespace contains built-in functions (such as abs(), cmp(), and others) as well as built-in exception names.

**Garbage Collection**

Garbage collector exposes the automatic garbage collector, which is the foundation of Python's memory management system. The module contains methods for controlling the collector's activities and for examining recognized objects that are either in the process of being collected or that are caught in reference cycles and are unable to be released.

**Python XML Parser**

Programmers can construct programs that function with any operating system and/or language for development because XML is open source and portable.

What is XML exactly? An extended markup language XML is a markup language, just as HTML and SGML.

This is recommended by the World Wide Web Consortium and is provided as an open standard.

For handling small to medium volumes of data, XML is particularly useful as it doesn't require a SQL- based backend.

XML Parser Architectures and APIs A collection of straightforward yet functional XML interfaces is provided by the Python standard library.

The two most basic and popular APIs for XML data are the SAX and DOM interfaces.

Basic XML SAX API: In this case, you first let the parser to go through the text while setting up call- backs for significant events.

If your documents are large or you have memory limitations, this feature is helpful since it parses the file as it reads it from disc and never stores the entire file in memory.

Document Object Model's DOM API To represent every feature of an XML document, the entire file

is read into memory and stored using a hierarchical tree structure. The World Wide Web Consortium suggests doing this.

It is evident that SAX cannot process data as quickly as DOM when working with large files. However, if you use DOM exclusively, it could cause a significant resource drain, particularly if you use it for a lot of little files.

Whereas SAX is read-only, DOM allows changes to be made to the XML file. Since these two different APIs complement each other so well, there is no reason why you cannot use them both for large-scale applications.

**Python Web Frameworks**

A web framework is a code library that makes it easier for developers to create scalable, reliable, and maintainable web applications.

**What is the worth of web frameworks?**

The expertise that programmers have gained over the previous 20 years while developing websites and online-based applications is included in web frameworks. Frameworks allow additional developers who are familiar with the framework to construct and maintain the application more effectively by facilitating code reuse for regular HTTP operations and project structuring.

typical features of web frameworks

Frameworks provide functionality to do common activities required to run web applications, either directly in the source code or through extensions.

These common functions include:

* URL forwarding
* HTML, XML, JSON, and other output format templating
* Inappropriate database use
* Defense against many attacks, including Cross-Site Request Forgery (CSRF)
* Session storage and retrieval

Not every web framework contains the code required to support the functionalities stated above. Depending on where they fall on the spectrum, frameworks can implement a single use case or provide every known web framework capability to every developer. There are frameworks that follow the "batteries-included" approach, meaning they contain everything that is useful, and others that include

a minimum core package that may be extended by other packages.

**Web framework comparison**

Additionally, different Python web frameworks, templating engines, and object libraries are being used to construct the same online application under the compare-python-web-frameworks repository.

**Web framework resources**

* Learning how to use one or more web frameworks requires an understanding of their underlying code.
* Frameworks is a short, incredibly well-made video that shows you how to choose a web framework. The author holds certain particular opinions about what should be included in a framework.

I still largely agree, even though I've learned that, when done well, sessions and database ORMs may be a useful part of a framework for corresponding web apps made with Django, Flash, and Pyramid, the three main Python frameworks.

What is a web framework? explains in depth the definition of web frameworks and its connection to web servers.

* Background data and code comparisons for related online applications made in these three popular Python frameworks—Django, Flash, and Pyramid—are included when choosing a web framework for Python programming.
* This interesting blog post examines the code complexity of different Python web frameworks by providing visualisations based on their code bases.
* The Python web framework benchmarks use the extraction of data from a database, its display in a template, and the encoding of an object into JSON and its subsequent response as ways to measure a framework's responsiveness. Even though the results were inconclusive, reading about the product is still interesting.

What are your favorite web frameworks, and which ones do you use? is a language-neutral Reddit discussion on web frameworks. It's interesting to look at what aspects of web frameworks for other programming languages Python programmers like and dislike differ from those of the main Python frameworks.

Which all-purpose Python web frameworks are the most recommended for use in production, based on user votes on this Q&A website? SMore important than the votes is the list of many frameworks from which Python developers can select.

**Web frameworks learning checklist**

* Choose a well-liked Python web framework and stick with it (Flask or Django are recommended). When you're first starting out, it's better to focus on mastering one framework rather than trying to learn every framework at once.
* Adhere to the comprehensive instructions located in the resource links on the page of the framework.
* Look at open-source projects made with the framework of your choice so you can reuse part of the code in your application.
* Before proceeding with the deployment phase to make your web application accessible online, create the first, most basic version of it.

1. **SYSTEM DESIGN**
   1. **UML (Unified Modeling Language)**

Unified Modeling Language which is employed in software engineering that is object- oriented. This rich language, which is intended to give a standard method for representing a system's architecture, can be used to represent an application's behavior, structures, and even business processes, while being mostly employed in software engineering.

Grady Booch, Ivar Jacobson, and James Rumbaugh of Rational Software designed and developed it from 1994 to 1995, and they continued to lead development until 1996. The Object Management Group (OMG) approved it as a standard in 1997, and this group has been in charge of its management ever since. The International Organization for Standardization (ISO) also recognized the Unified Modeling Language in 2000 as an established ISO standard.

* + 1. **Class Diagram**

Class diagrams are probably the most used UML diagram type. These are the basic building blocks of any object-oriented solution1. It shows the classes in the system, the properties and behavior of each class, and the relationships between each class. In most modeling tools, a class has three parts: the name at the top, the properties in the middle, and the operations or methods at the bottom.

* + 1. **Component Diagram**

The Component diagram shows the relationship between software system components. These are often used when dealing with complex processes that have multiple components. Components communicate with each other using interfaces. These interfaces are connected by connectors.

* + 1. **Deployment Diagram**

A deployment diagram shows the hardware in a system and the software on that hardware. Deployment diagrams are useful when a software solution is deployed to multiple computers, each with its own configuration.

* + 1. **Object Diagram**

Class diagrams and object diagrams are quite similar. Another name for object diagrams is instance diagrams. They use real-world examples, but as class diagrams, they also illustrate the link between items. They provide as a visual representation of a system at a specific moment in time.

* + 1. **Use Case Diagram**

Ucase diagrams, the most popular sort of behavioral UML diagram, provide a visual summary of the many players in a system, the various functions they require, and the interactions between these various activities. It’s a wonderful place to start any project discussion since it makes it simple to identify the key players and system operations.

* + 1. **Activity Diagram**

Activity diagrams provide a graphical representation of a workflow. They can be used to the operational workflow of any system component as well as business workflow. State machine diagrams can occasionally be substituted by activity diagrams.

* + 1. **State Machine Diagram**

Activity diagrams and state machine diagrams are comparable, although there are some differences in notation and usage. They might also be referred to as start chart diagrams or state diagrams. These are highly helpful in characterizing the actions of things whose behavior varies based on their current state.

* + 1. **Sequence Diagram**

UML sequence diagrams display the relationships between objects and the sequence in which they occur. It is noteworthy that illustrates the interactions for a specific case. Interactions are displayed as arrows, and the processes are shown vertically.

* + 1. **Collaboration Diagram**

It is similar to sequence diagrams but the focus is on messages passed between objects. The same information can be represented using a sequence diagram and different objects.

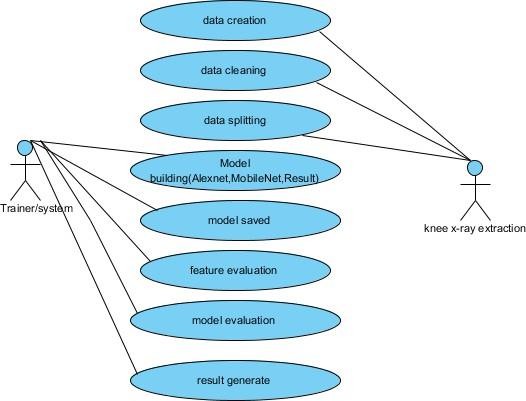
* 1. **List of UML Notations**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **SYMBOL NAME** | **SYMBOL** | **DESCRIPTION** |
| 1 | Class |  | Classes represent a collection of similar entities grouped together. |
| 2 | Association |  | Association represents a static relationship between classes. |
| 3 | Aggregation |  | Aggregation is a form of association. It aggregates several classes into single class. |
| 4 | Actor | Actor | Actors are the users of the system and other external entity that react with the system. |
| 5 | Use Case |  | A use case is an interaction between the system and the external environment. |
| 6 | Relation (Uses) |  | It is used for additional process communication. |
| 7 | Communication |  | It is the communication between various use cases. |
| 8 | State |  | It represents the state of a process. Each state goes through various flows. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | Initial State |  | | | | | It represents the initial state of the object. |
| 10 | Final State |  | | | | | It represents the final state of the object. |
| 11 | Control Flow |  | | | | | It represents the various control flow between the states. |
| 12 | Decision Box |  | | | | | It represents the decision making process from a constraint. |
| 13 | Component |  | | | | | Components represent the physical components used in the system. |
|  | |  | |  |
|  |  |  |  |
|  |  |  | **Component** |
|  |  |  |  |
|  | |  | |  |
|  | |
| 14 | Node | **Node** | | | | | Deployment diagrams use the nodes for representing physical modules, which is a collection of components. |
| 15 | Data Process/State |  | | | | | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 16 | External Entity |  | | | | | It represent anyexternal entity such as keyboard, sensors etc. |
| 17 | Transition |  | | | | | It represents anycommunication that occurs between the processes. |
| 18 | Object Lifeline |  | | | | | Object lifelines represents the vertical dimension that objects communicates. |
| 19 | Message |  | | | | | It represents the messages exchanged. |

* 1. **UML Diagrams**
     1. **Use case diagram**

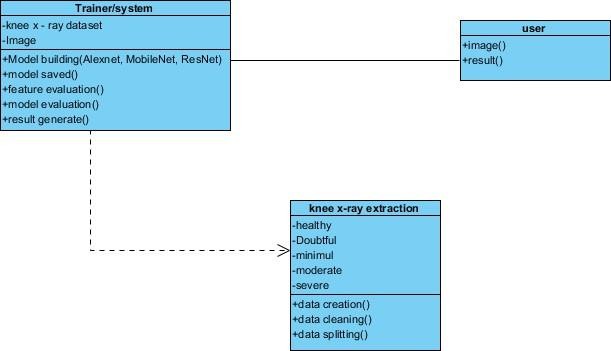
A use case diagram is a type of behavioral diagram generated from a Unified Modeling Language (UML) use-case study. Its goal is to provide a graphical overview of the functionality that a system provides by showing the actors, their goals (expressed as use cases), and any interdependencies among the use cases. The principal aim of a use case diagram is to denote which actors are served by which system functionalities. It is possible to illustrate the roles that the actors in the system play.



# Fig:8.3.1 Use case diagram

* + 1. **Class diagram**

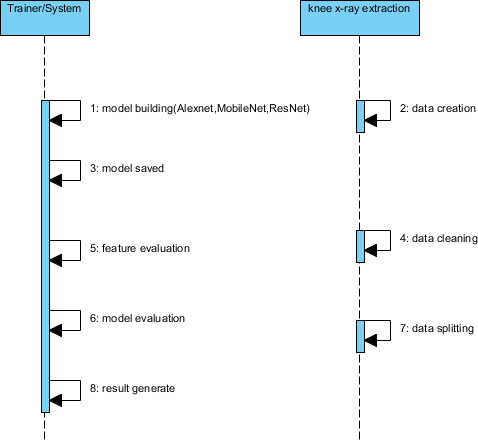
Class diagrams are probably the most used UML diagram type. These are the basic building blocks of any object-oriented solution1. It shows the classes in the system, the properties and behavior of each class, and the relationships between each class. In most modeling tools, a class has three parts: the name at the top, the properties in the middle, and the operations or methods at the bottom.



# Fig:8.3.2 Class Diagram

* + 1. **Sequence Diagram**

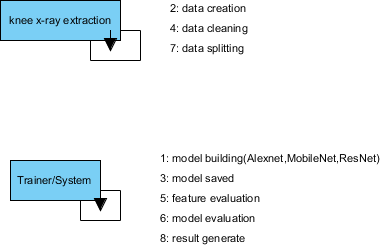
A sequence diagram, a kind of interaction diagram used in the Unified Modeling Language (UML), shows the connections and sequence of events across processes. It is incorporated into the design of the message sequence chart. Sequence diagrams are sometimes referred to as event diagrams, event scenarios, and timing diagrams.



# Fig:8.3.3 Sequence Diagram

* + 1. **Collaboration diagram**

The order in which the methods are invoked is indicated by a numbering scheme in the collaboration diagram that follows. The sequence in which the methods are invoked is indicated by the number. The order management system used to describe the collaboration diagram is the same. The method calls are analogous to a sequence diagram. However, while the sequence diagram merely describes the item arrangement, the collaboration diagram shows it.



# Fig:8.3.4 Collaboration diagram

* + 1. **Deployment Diagram**

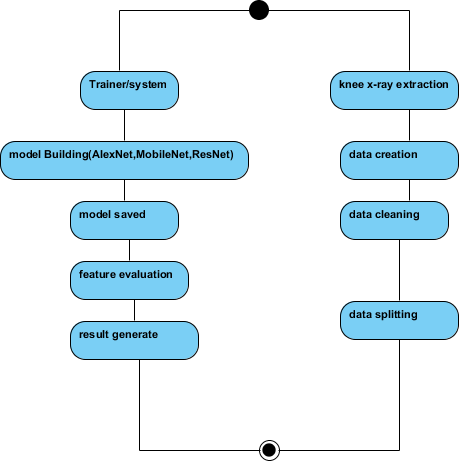
A deployment diagram shows the hardware in a system and the software on that hardware. Deployment diagrams are useful when a software solution is deployed to multiple computers, each with its own configuration.



# Fig:8.3.5 Deployment diagram

* + 1. **Activity Diagram**

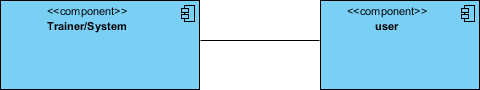
Activity diagrams provide a graphical representation of a workflow. They can be used to the operational workflow of any system component as well as business workflow. State machine diagrams can occasionally be substituted by activity diagrams.



# Fig:8.3.6 Activity Diagram

* + 1. **Component Diagram**

The Component diagram shows the relationship between software system components. These are often used when dealing with complex processes that have multiple components. Components communicate with each other using interfaces. These interfaces are connected by connectors.

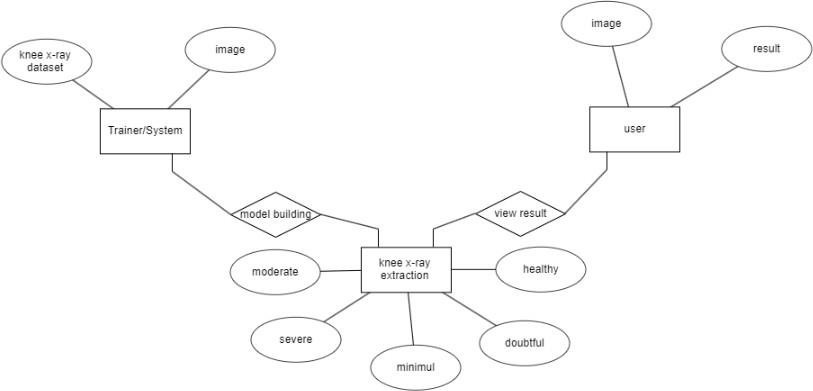


# Fig:8.3.7 Component Diagram

* + 1. **ER Diagram**

The structure of a database is shown using an Entity-Relationship Model (ER model), which uses an Entity Relationship Diagram (ER Diagram). An ER model, which is a database design or blueprint, can then be put into practice as a database. The main components of the E-R model are the entity set and the relationship set.

The relationships between entity sets are depicted in an ER diagram. A group of linked entities, each of which might have properties, is called an entity set. Since a table or one of its characteristics is considered an entity in a DBMS, the ER diagram shows the relationships between tables and their attributes to show the logical structure of the database as a whole.

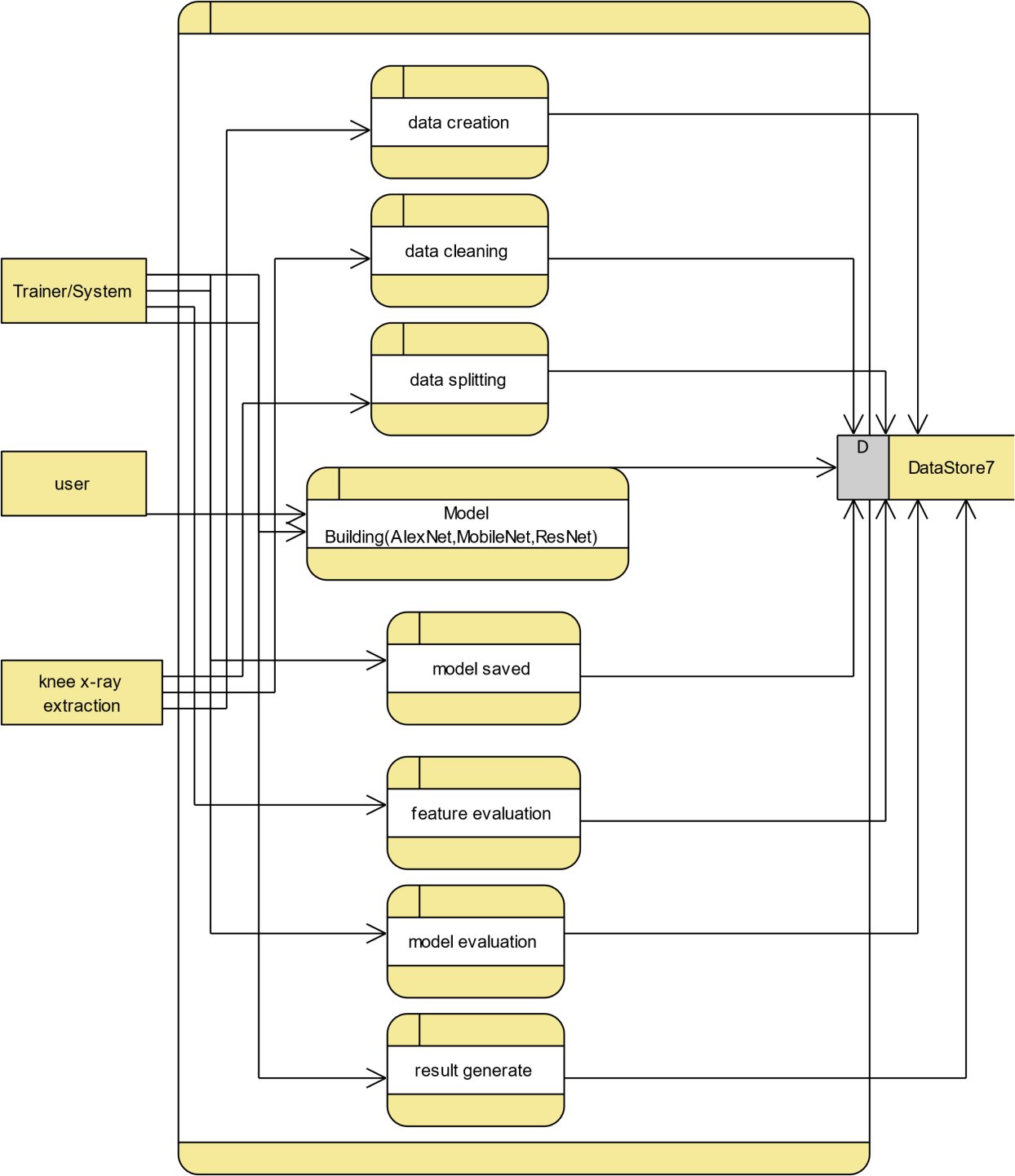


# Fig:8.3.8 ER Diagram

* + 1. **DFD diagram**

A data flow diagram (DFD) is a common tool for visualizing how information moves through a system. A clear and concise DFD can graphically depict a large portion of the system requirements. Either manually, automatically, or both can accomplish it. It shows where data is stored, how it is modified, and how it enters and leaves the system. A DFD is used to show the boundaries and overall scope of a system. It can be used as a means of communication between a systems analyst and any system user and forms the cornerstone of system redesign.

# Fig:8.3.9 DFD diagram



1. **TESTING**

Testing is done in order to find errors. Finding any errors or shortcomings in a piece of work is the process of testing it. It provides a means of analyzing the functionality of individual components, assemblies, subassemblies, and/or finished goods. It is the process of testing software to ensure that it fits criteria and user expectations without malfunctioning in an undesired way. There are multiple types of exams. Each sort of test caters to a specific testing requirement.

**9.3 TYPES OF TESTS**

* + 1. **Unit testing**

Creating test cases for unit testing guarantees that program inputs produce valid outputs and that the fundamental logic of the program is operating as intended. Checking all decision branches and the internal code flow is crucial. It is the testing of each software component that makes up the program. It is completed after every separate unit is finished, prior to integration.

This is an invasive structural test that requires knowledge of the construction process. Unit tests investigate a specific configuration of a system, application, or business process and perform basic tests at the component level. Unit tests provide assurance that every individual path of a business process follows the specified guidelines exactly and has clearly defined inputs and outputs.

* + 1. **Integration testing**

This process examines if newly combined software components actually function as a single program. Testing is event-driven and concentrates more on the core output of fields or screens. Integration tests show that the combination of the components is correct and consistent, even if the individual components passed unit testing. Integration testing is specifically intended to draw attention to problems that arise when integrating components.

* + 1. **Functional Testing**

Functional tests provide systematic evidence that the functions under test are available in compliance with user manuals, system documentation, and business and technical requirements.

Functional testing focuses on the following domains:

Valid Input : Classes of acknowledged valid input must be approved. Invalid Input : It is necessary to reject the defined kinds of invalid input. Functions : Utilizing the functions on this list is required.

Output : The stated application output kinds must be practiced. Systems/Procedures : Calling interdependent systems or processes is required.

Requirements, significant features, or distinctive test cases are the main considerations in the organization and preparation of functional tests. Additionally, testing needs to consider business process flows, defined procedures, and following processes in addition to systematically covering data fields. Before functional testing is completed, further tests are discovered and the value of the ones that already exist is evaluated.

* 1. **SYSTEM TEST**

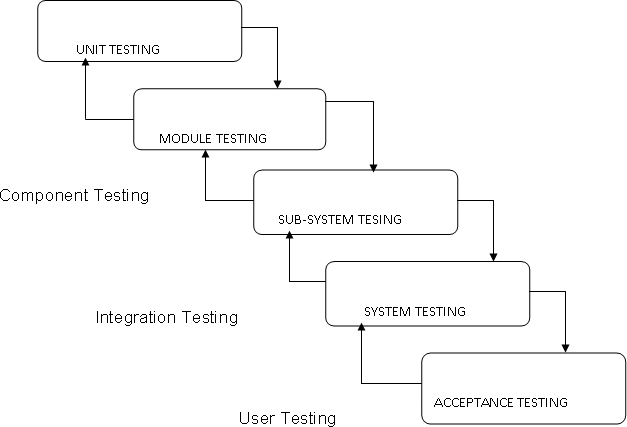
System testing ensures that all of the integrated software system's components meet the required standards. It assesses a configuration to produce known and expected results. One example of a system test is a configuration-oriented system integration test. Process flows and descriptions serve as the foundation for system testing, which focuses on pre-driven integration points and links.

* + 1. **White Box Testing**

In white box testing, the software tester is someone who understands the inner workings, language, and structure of the program, or who at least is aware of its intended functionality. There's a rationale. Its goal is to test locations that are off-limits to users on a black box level.

* + 1. **Black Box Testing**

Software tested in a "black box" is one that is tested without the tester being aware of the inner workings, architecture, or language of the module being tested. Like the majority of other test kinds, black box tests need to be developed from a clear source document, like a requirements or specification document. In this kind of testing, the program under test is treated as if it were a black box. You cannot "look" inside of it. The test creates inputs and responds to outputs without considering the operation of the program.



**Fig:9.2.2 Testing Cycle**

* + 1. **Unit Testing**

Although it is common for coding and unit testing to be completed as two distinct processes, unit testing is usually carried out as a part of a combined code and testing process.

**Test strategy and methodology**

Both thorough functional tests and manual field testing will be conducted.

**Test goals**

* + - 1. Every field entry needs to work correctly.
      2. To activate the pages, click on the corresponding link.
      3. It is not acceptable to have delays in the entering screen, messages, or responses.

**Features that need to be tested are**

* + - 1. Verify that all links point to the correct page
      2. Disallow duplicate entries
      3. Verify that entries adhere to the correct format.
    1. **Integration Testing**

"Software integration testing" is the process of gradually evaluating two or more integrated software components on a single platform in order to cause errors that result from interface defects.

An integration test's objective is to verify that software programs or parts, like those in a software system or, at a higher level, software programs utilized by an entire company, operate seamlessly together.

* + 1. **Acceptance Testing**

User approval the testing stage of every project is essential and demands the active participation of the end user. Additionally, it ensures that the system meets the functional requirements.

**Results of the exams:** The test situations listed above were all successful. No flaws could be discovered.

1. **Results**
   1. **Evaluation metrics**

**Accuracy:**

The measurement of accurate estimates is accuracy. Regarding FP, TP, TN, and FN, accuracy is



**Precision:**

Precision is how many images predicted as positive are positive.



**Recall:**

The recall is the ratio of images correctly identified as positive.



**F1 Score:**

The F1 score or F Measure is used to measure the accuracy of a model. The greater the F1 score, the better the performance of our model.



**Mean Square Error:**

Mean Squared Error (MSE) is the mean or average of the square of the difference between actual and estimated values. For a dataset of n images, given the actual label and predicted label, the MSE is defined as

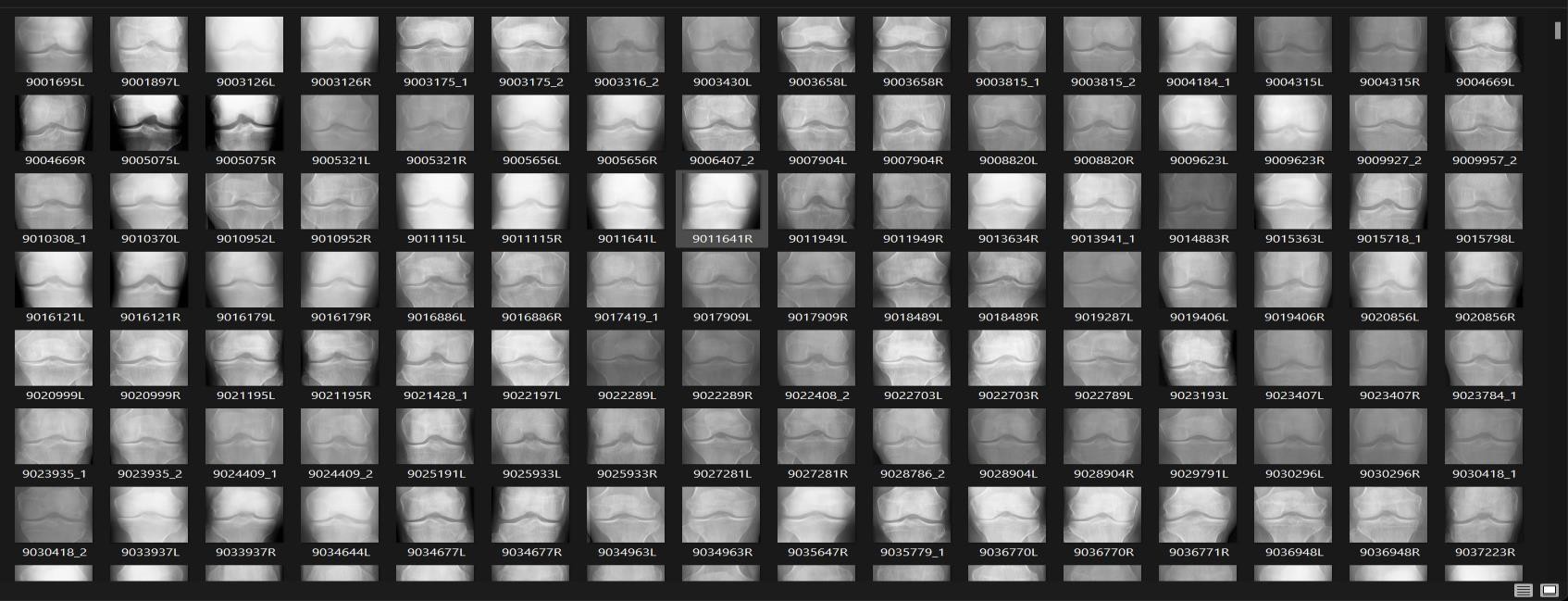


**ROC:**

The Receiver Operating Characteristics Curve (ROC) measures how accurately the model can differentiate between different classes, while Area Under the Curve (AUC) measures the entire 2-dimensional area underneath the ROC curve.

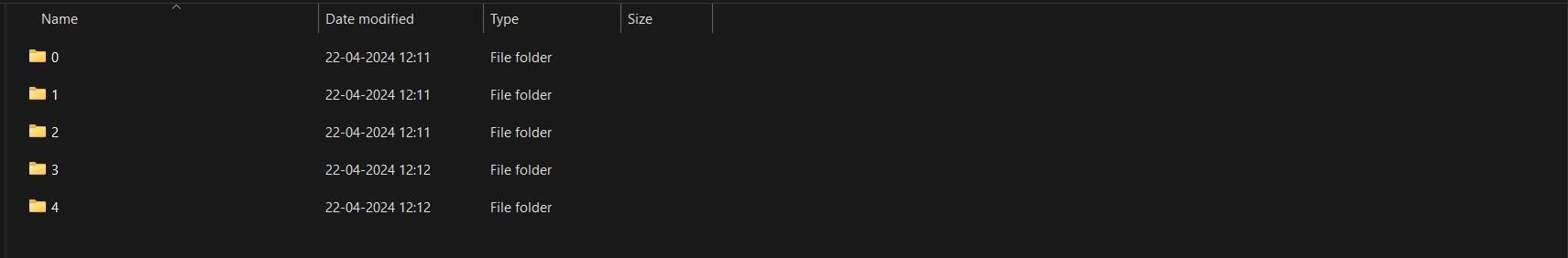
* 1. **Screenshots View data set**

The dataset used to train the model.

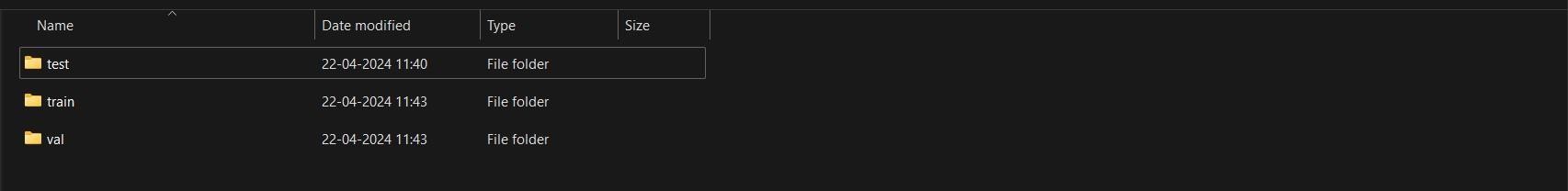


# Fig:10.2.1 Dataset

**Data set of different KL-Grades.**



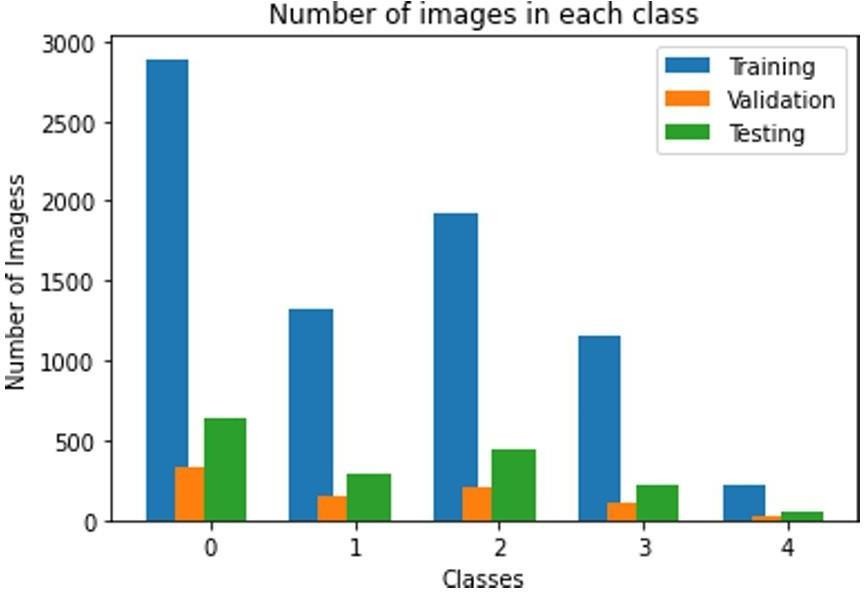
**Fig:10.2.2 KL-Grades different levels Splitting of the data set for training, validation & testing.**



**Fig:10.2.3 Splitting of the data**

**Splitting of Dataset**

The dataset splits into training, validation & testing and it is shown in below bar graph.



# Fig:10.2.4 Splitting of Dataset

**Attributes of the dataset:**

This dataset contains knee X-ray data for both knee joint detection and knee KL grading. The Grade descriptions are as follows:

* Grade 0: Healthy knee image.
* Grade 1 (Doubtful): Doubtful joint narrowing with possible osteophytic lipping
* Grade 2 (Minimal): Definite presence of osteophytes and possible joint space narrowing
* Grade 3 (Moderate): Multiple osteophytes, definite joint space narrowing, with mild sclerosis.
* Grade 4 (Severe): Large osteophytes, significant joint narrowing, and severe sclerosis.

We have used the 4893 men and women for the gathering of the Knee X-ray images used in our study are based on this dataset of the total of 9786 X-ray images. The dataset was split into the training, testing, and validation set with a 7.5: 1.5: 1 ratio.

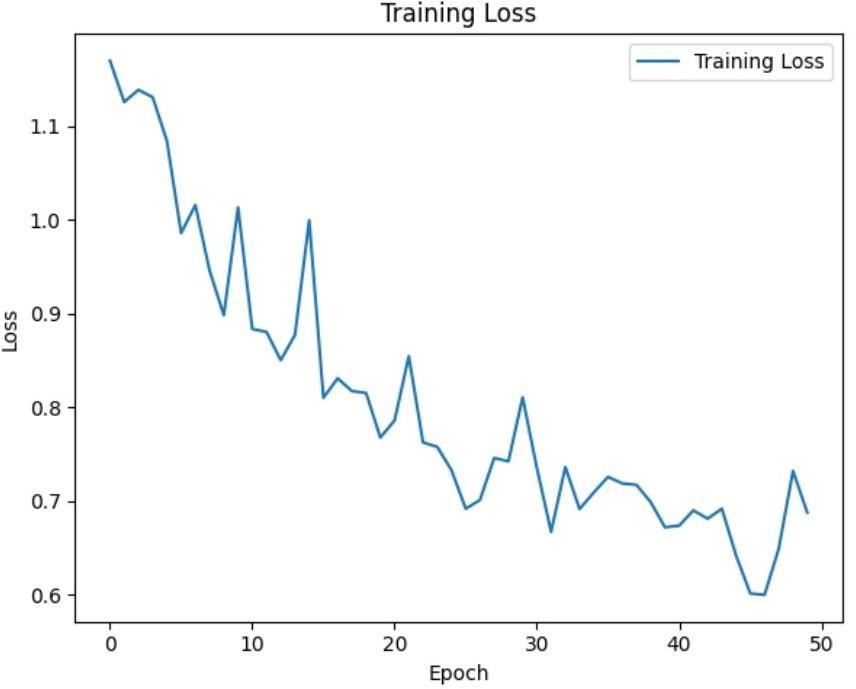
**Data set splitting for the model:**

* + 1. **Training:** The data is used for the training of the model is 7334 X-ray images.
    2. **Testing:** The data is used for the testing & evaluating of the model is 1483 X-ray images.
    3. **Validation:** The data used for the validation of the model is 969 X-ray images.

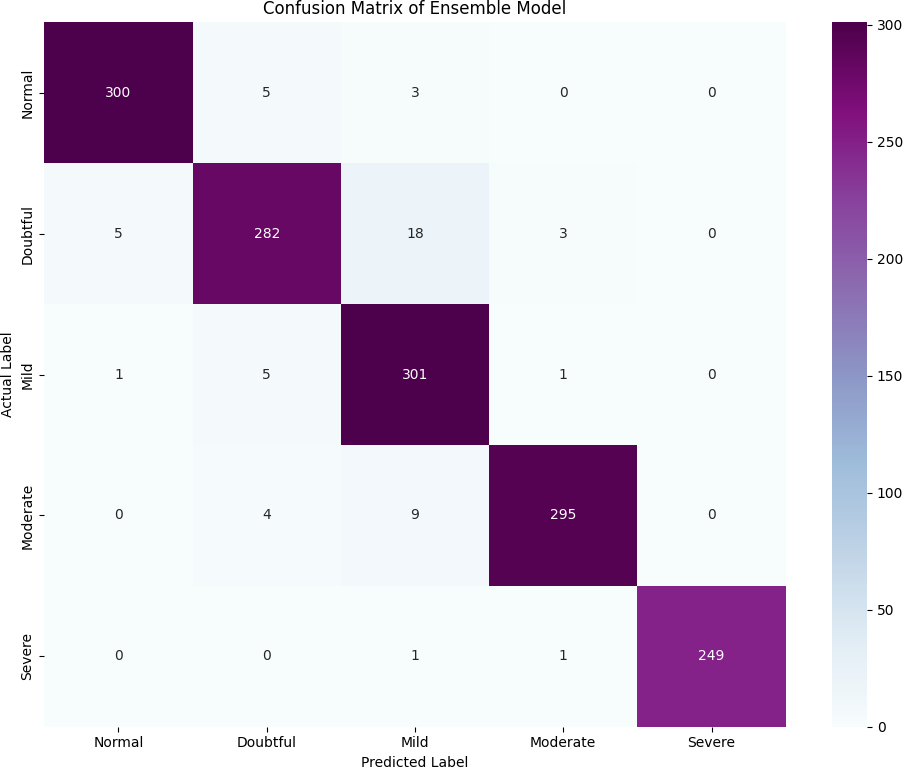
**Model Training & Evalution Results:**



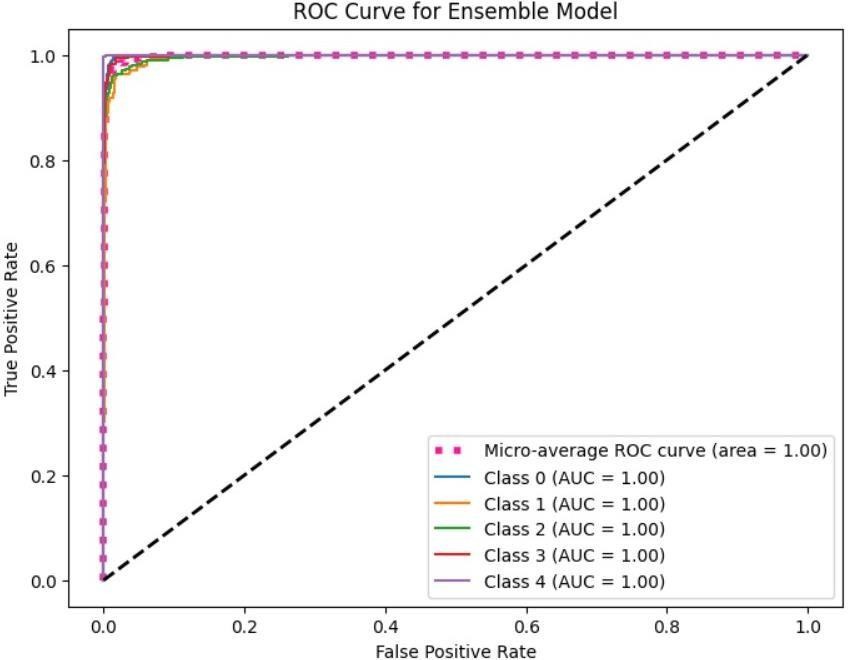
# Fig:10.2.5 Model Training Accuracy



# Fig:10.2.6 Model Training Loss



# Fig:10.2.7 Confusion Matrix of the Model

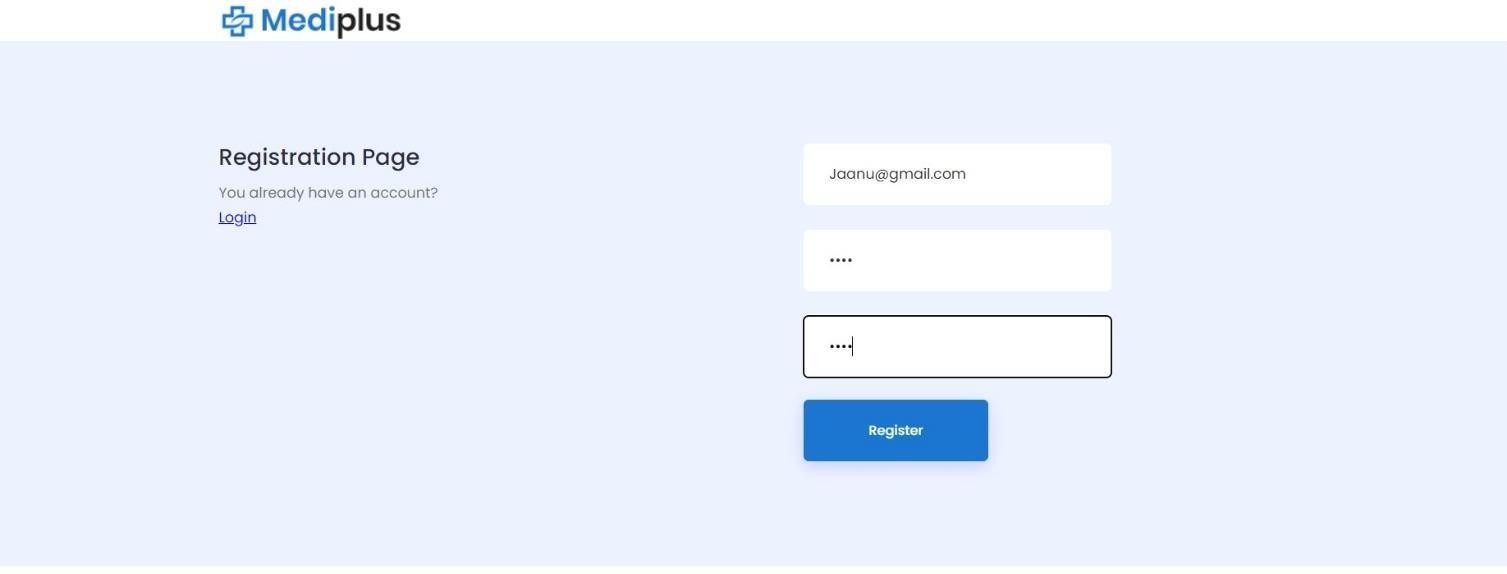


**Fig:10.2.8 ROC & AUC of the Mode**

**Web Application Interfaces:**



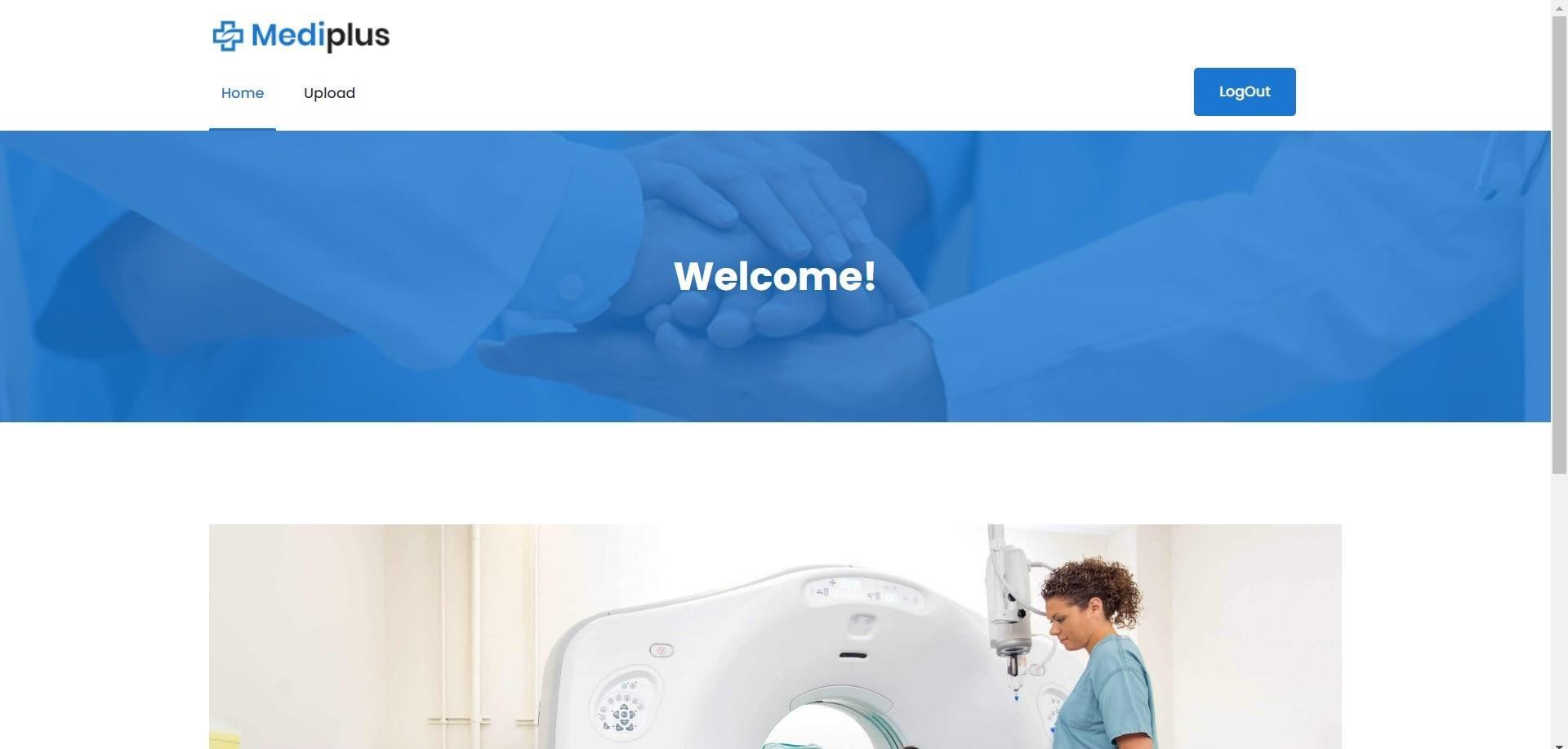
# Fig:10.2.9 Website Interface

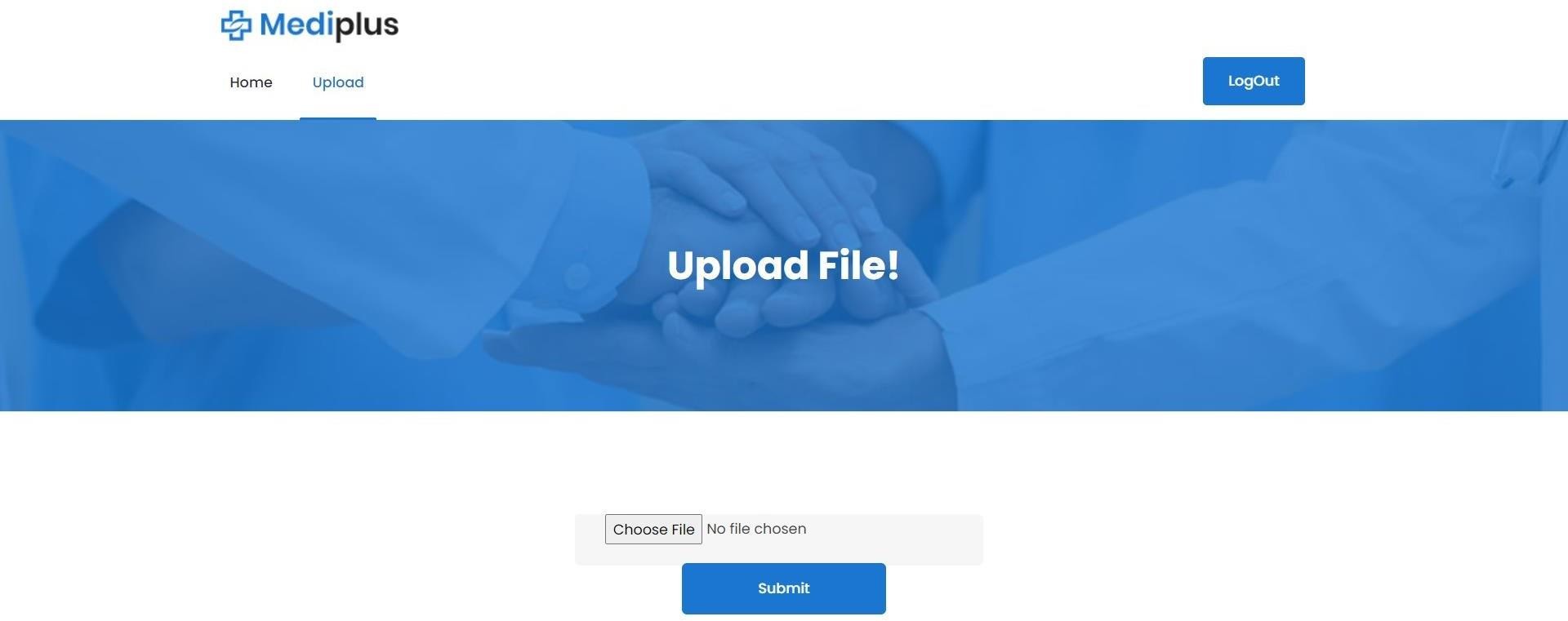


# Fig:10.2.10 Registration page

# Fig:10.2.11 Login page

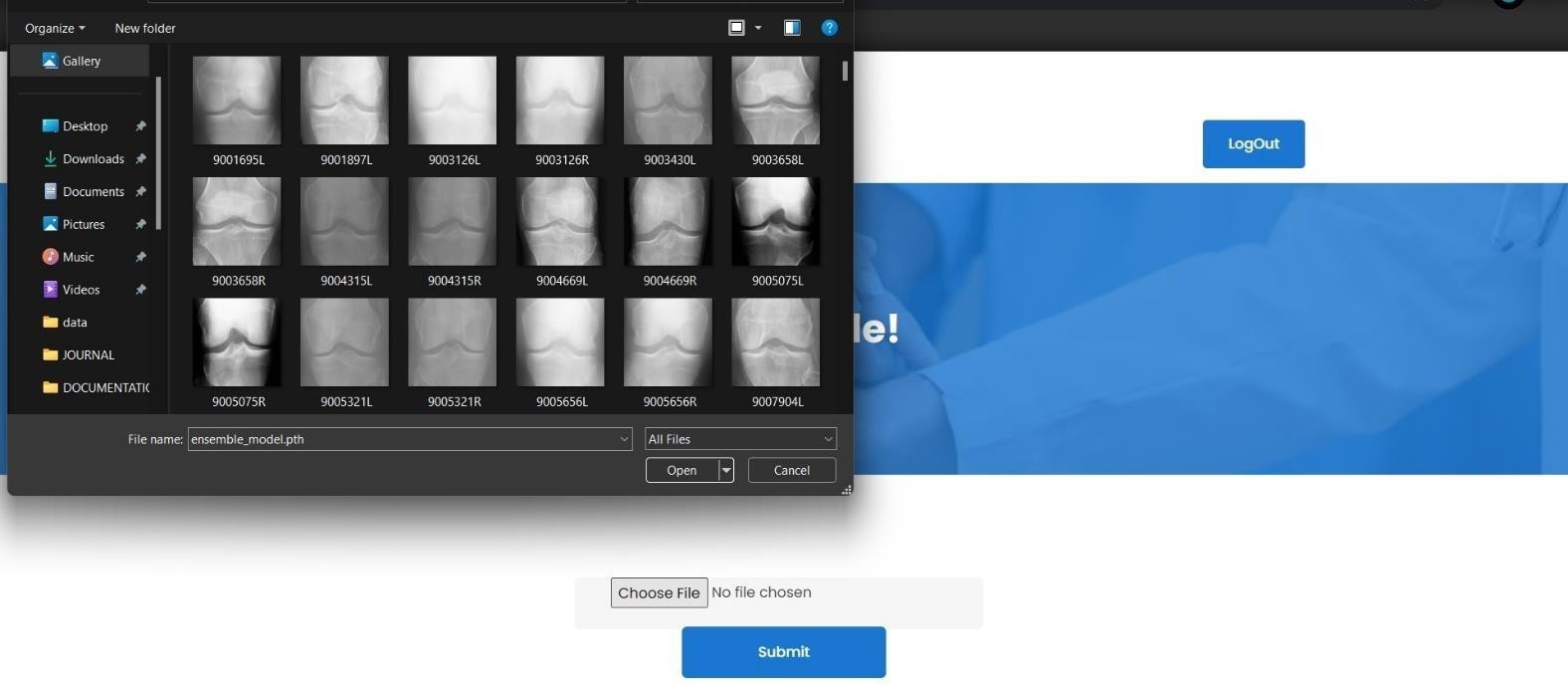
# Fig:10.2.12 Home page

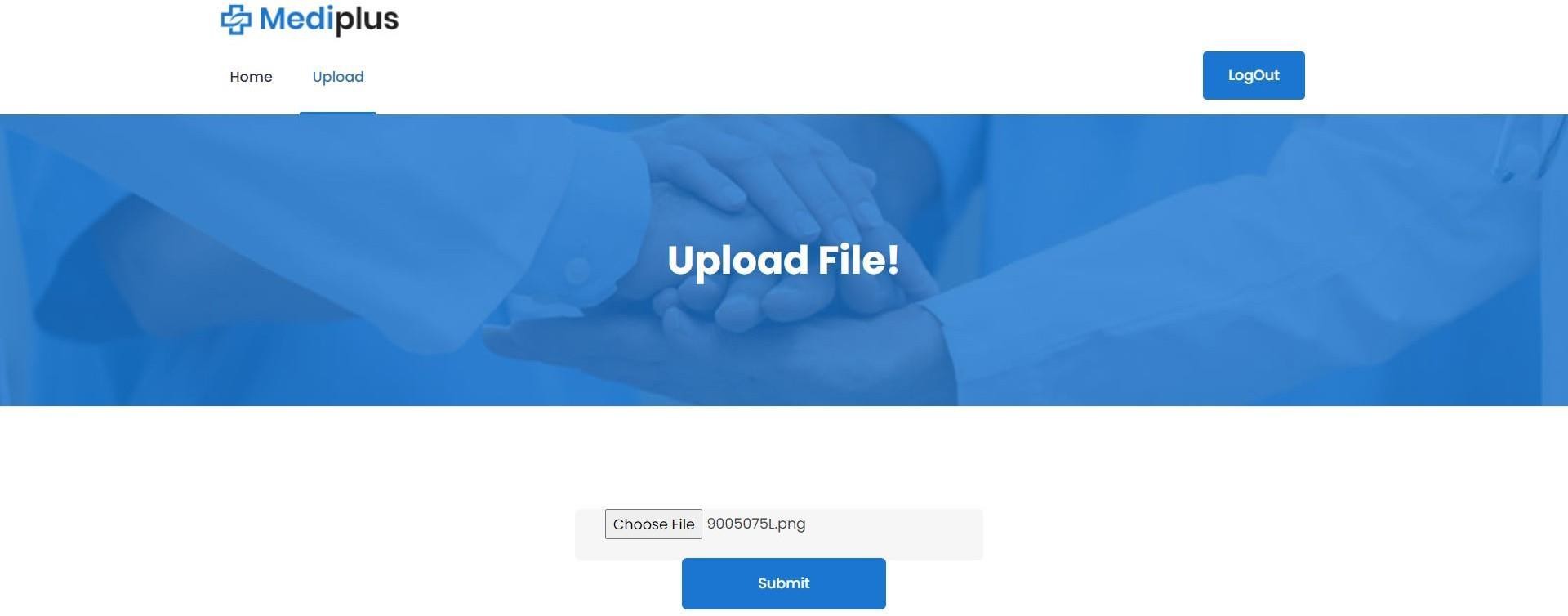




# Fig:10.2.13 Upload File

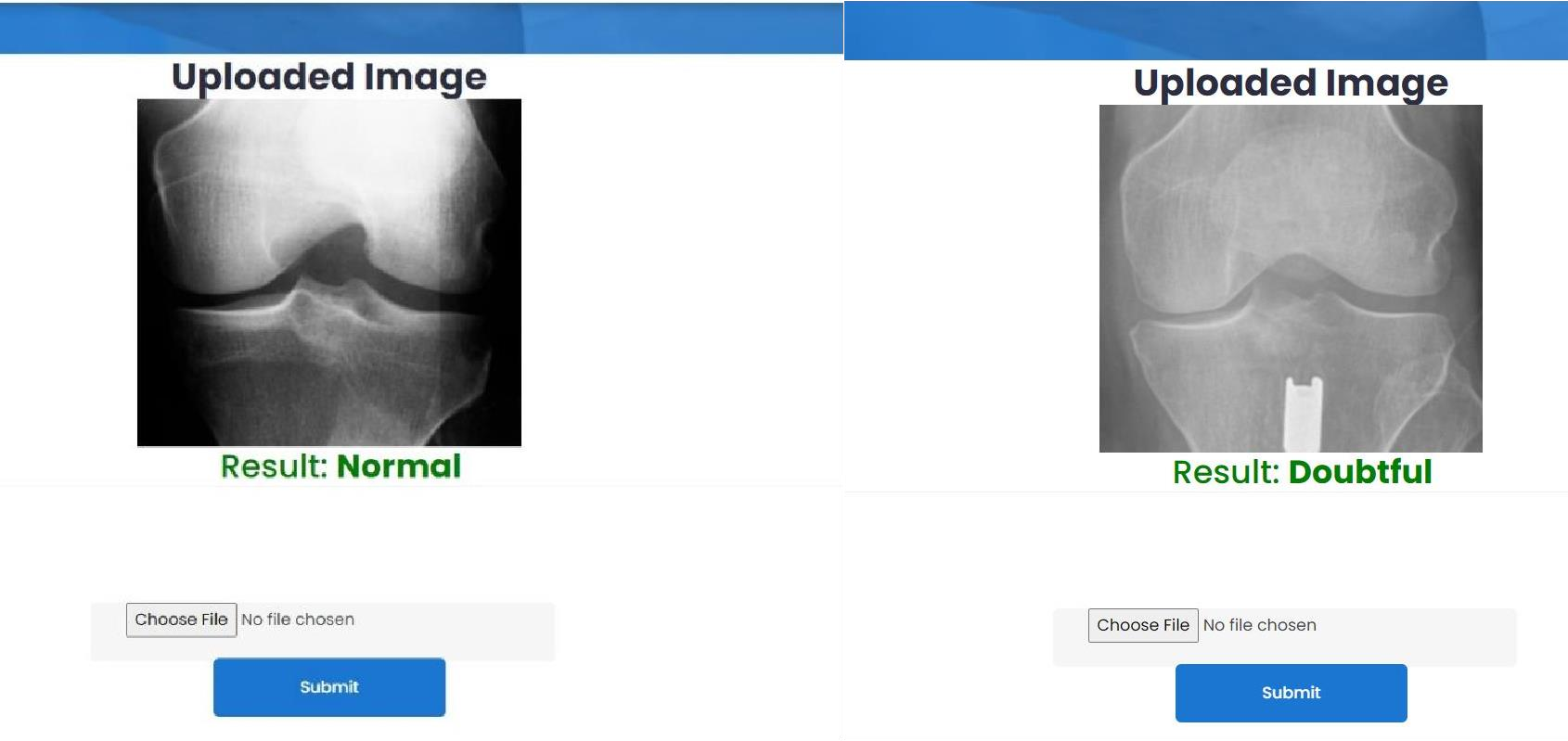
# Fig:10.2.14 Choose file



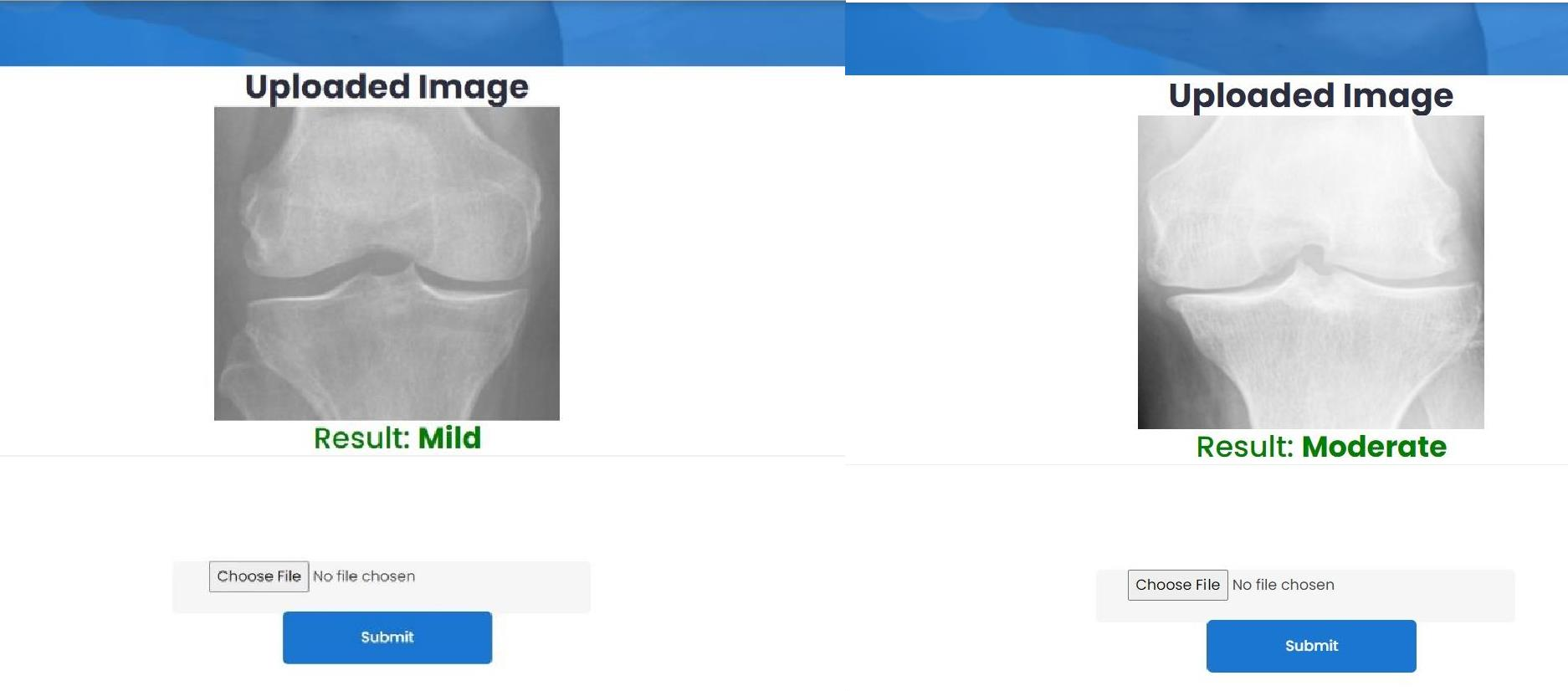


# Fig:10.2.15 Select file

**Graph**

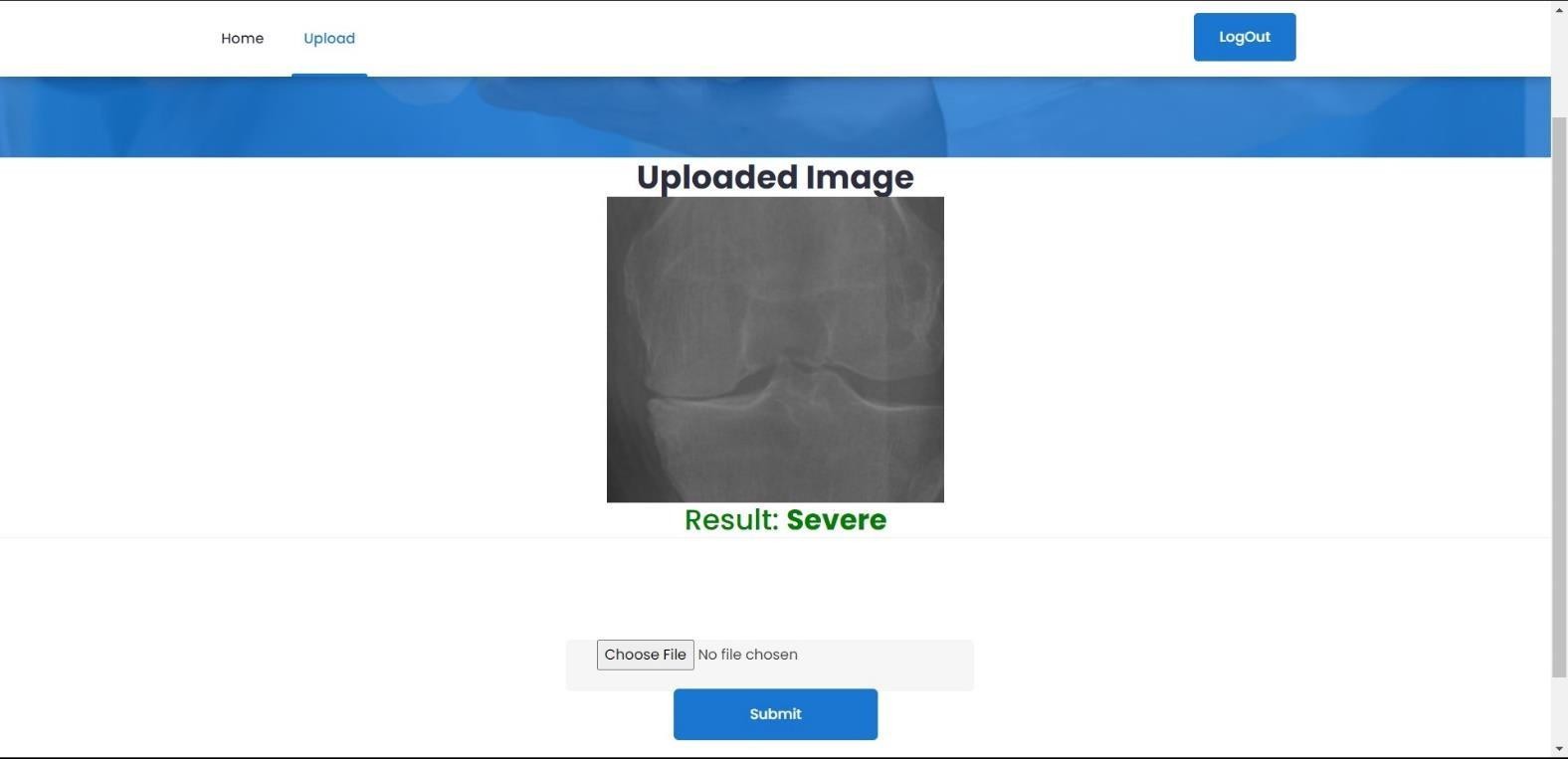


**Fig:10.2.16 Normal Fig:10.2.17 Doubtful**



**Fig:10.2.18 Mild Fig:10.2.19 Moderate**

**Fig:10.2.20**



1. **CONCLUSION**

In summarizing, the utilization of advanced learning techniques, notably AlexNet, MobileNet, and ResNet, demonstrates a strong commitment to advancing the detection of knee injuries in X-ray assessments. Despite achieving commendable accuracy in predictions, persistent challenges such as imbalanced data, speculation issues, and a lack of comprehensive classification studies remain. Addressing these challenges through enhanced deployment methods and strategic adjustments is crucial to bolster the reliability and effectiveness of deep learning models in clinically diagnosing knee injuries. Besides, coordinating highlights for adaptability and lightweight arrangement is fundamental to work with the far and wide acknowledgment of these models in clinical settings. Future examination attempts ought to focus on conquering these impediments to completely saddle the capability of profound learning in altering knee injury finding. in this manner propelling patient consideration and treatment results in muscular medication.

**12 FUTURE SCOPE**

In the realm of medical imaging and diagnosis, continuous advancements in technology pave the way for ever-improving methodologies. Building upon the foundation laid by the ensemble approach utilizing MobileNet, ResNet, and AlexNet architectures for knee osteoarthritis (KOA) classification, future enhancements could further elevate the efficacy and applicability of the model.

Firstly, incorporating newer CNN architectures that have emerged since the study's inception can expand the model's capacity to extract intricate patterns from knee joint X-ray images. Keeping abreast of the latest developments in deep learning ensures that the model remains at the forefront of innovation, potentially enhancing its accuracy and generalization capabilities.

Moreover, diversifying the dataset to include a broader range of demographic and clinical characteristics can bolster the model's robustness and real-world relevance. By encompassing a more comprehensive representation of patients with varying degrees of KOA severity, the model can better adapt to the complexities and nuances inherent in clinical practice.

Furthermore, conducting rigorous clinical validation studies in diverse healthcare settings is imperative to assess the model's real-world performance and impact on patient outcomes. Collaborating with clinicians and healthcare professionals ensures that the model aligns with clinical workflows and addresses unmet needs in KOA diagnosis and management.

By embracing these future enhancements, the ensemble approach to KOA classification can continue to advance the frontiers of medical image analysis, ultimately improving diagnostic processes and patient care in the fight against osteoarthritis.

**13.REFERENCES**

References for the Project Development are taken from the following Books and Web Sites:

1. Y. Badshah, M. Shabbir, H. Hayat, Z. Fatima, A. Burki, S. Khan, and S. U. Rehman, ‘‘Genetic markers of osteoarthritis: Early diagnosis in susceptible Pakistani population,” J. Orthopaedic Surgery Res., vol. 16, no. 1, pp. 1–8, Dec. 2021.
2. Y. X. Teoh, K. W. Lai, J. Usman, S. L. Goh, H. Mohafez, K. Hasikin, P. Qian, Y. Jiang, Y. Zhang, and S. Dhanalakshmi, ‘‘Discovering knee osteoarthritis imaging features for diagnosis and prognosis: Review of manual imaging grading and machine learning approaches,’’ J. Healthcare Eng., vol. 2022, pp. 1–19, Feb. 2022.
3. A. Tiulpin and S. Saarakkala, ‘‘Automatic grading of individual knee osteoarthritis features in plain radiographs using deep convolutional neural networks,’’ Diagnostics, vol. 10, no. 11, p. 932, Nov. 2020.
4. P. Chen. (2018) Knee Osteoarthritis Severity Grading Dataset. [Online].Available: https://data.mendeley.com/datasets/56rmx5bjcr/1/
5. D. Saini, T. Chand, D. K. Chouhan, and M. Prakash, ‘‘A comparative analysis of automatic classification and grading methods for knee osteoarthritis focussing on X-ray images,’’ Biocybernetics Biomed. Eng., vol. 41, no. 2, pp. 419–444, Apr. 2021.
6. C. W. Yong, K. Teo, B. P. Murphy, Y. C. Hum, Y. K. Tee, K. Xia, and K. W. Lai, ‘‘Knee osteoarthritis severity classification with ordinal regression module,’’ Multimedia Tools Appl., vol. 81, pp. 41497–41509, Jan. 2021.
7. P. Chen, L. Gao, X. Shi, K. Allen, and L. Yang, ‘‘Fully automatic knee osteoarthritis severity grading using deep neural networks with a novel ordinal loss,’’ Computerized Med. Imag. Graph., vol. 75, pp. 84–92, Jul. 2019.
8. T. Tariq, Z. Suhail, and Z. Nawaz, ‘‘Machine learning approaches for the classification of knee osteoarthritis,’’ in Proc. 3rd Int. Conf. Electr., Comput., Commun. Mechatronics Eng. (ICECCME), 2023.
9. B. Norman, V. Pedoia, A. Noworolski, T. M. Link, and S. Majumdar, ‘‘Applying densely connected convolutional neural networks for staging osteoarthritis severity from plain radiographs,’’ J. Digit. Imag., vol. 32, no. 3, pp. 471–477, Jun. 2019.
10. K. Leung, B. Zhang, J. Tan, Y. Shen, K. J. Geras, J. S. Babb, K. Cho,G. Chang, and C.

M. Deniz, ‘‘Prediction of total knee replacement and diagnosis of osteoarthritis by using deep learning on knee radio- graphs: Data from the osteoarthritis initiative,’’ Radiology, vol. 296, no. 3, pp. 584–593, Sep. 2020.

1. K. A. Thomas, L. Kidzinski, E. Halilaj, S. L. Fleming, G. R. Venkataraman, E. H. G. Oei, G. E. Gold, and S. L. Delp, ‘‘Automated classification of radiographic knee osteoarthritis severity using deep neural networks,’’ Radiol., Artif. Intell., vol. 2, no. 2, Mar. 2020, Art. no. e190065.
2. A. Brahim, R. Jennane, R. Riad, T. Janvier, L. Khedher, H. Toumi, and E. Lespessailles, ‘‘A decision support tool for early detection of knee OsteoArthritis using X-ray imaging and machine learning: Data from the OsteoArthritis initiative,’’ Computerized Med. Imag. Graph., vol. 73, pp. 11–18, Apr. 2019.
3. A. Mikhaylichenko and Y. Demyanenko, ‘‘Automatic grading of knee osteoarthritis from plain radiographs using densely connected convolu- tional networks,’’ in Proc. Int. Conf. Anal. Images, Social Netw. Texts. Cham, Switzerland: Springer, 2020, pp. 149–161.
4. A. Tiulpin, J. Thevenot, E. Rahtu, P. Lehenkari, and S. Saarakkala, ‘‘Automatic knee osteoarthritis diagnosis from plain radiographs: A deep learning-based approach,’’ Sci. Rep., vol. 8, no. 1, pp. 1–10, Jan. 2018.
5. X. Shi, W. Cao, and S. Raschka, ‘‘Deep neural networks for rank- consistent ordinal regression based on conditional probabilities,’’ 2021, arXiv:2111.08851.
6. M. B. Muhammad and M. Yeasin, ‘‘Eigen-CAM: Class activation map using principal components,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2020, pp. 1–7.
7. Jacob Gildenblat and Contributors. (2021). PyTorch Library for CAM Methods. [Online]. Available: https://github.com/jacobgil/pytorch-grad- cam
8. Y. Feng, J. Liu, H. Zhang, and D. Qiu, ‘‘Automated grading of knee osteoarthritis X- ray images based on attention mechanism,’’ in Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM), Dec. 2021, pp. 1927–1932.
9. W. Liu, T. Ge, L. Luo, H. Peng, X. Xu, Y. Chen, and Z. Zhuang, ‘‘A novel focal ordinal loss for assessment of knee osteoarthritis severity,’’ Neural Process. Lett., vol. 54, no. 6, pp. 5199– 5224, 2022.
10. B. Liu, J. Luo, and H. Huang, ‘‘Toward automatic quantification of knee osteoarthritis severity using improved faster R-CNN,’’ Int. J. Comput. Assist. Radiol. Surgery, vol. 15, no. 3,

pp. 457–466, Mar. 2020.