Detection of Osteoarthritis using Advance Segmentation in Radiographic Images

Project Team

Urooj Afridi	20P-0192
Alishan Khattak	21P-8046

Session 2020-2024

Supervised by

Dr. Muhammad Amin

Department of Computer Science

National University of Computer and Emerging Sciences
Peshawar, Pakistan

June, 2024

Student's Declaration

We declare that this project titled "*Detection of Osteoarthritis using Advance Segmentation in Radiographic Images*", submitted as requirement for the award of degree of Bachelors in Computer Science, does not contain any material previously submitted for a degree in any university; and that to the best of our knowledge, it does not contain any materials previously published or written by another person except where due reference is made in the text.

We understand that the management of Department of Computer Science, National University of Computer and Emerging Sciences, has a zero tolerance policy towards plagiarism. Therefore, We, as authors of the above-mentioned thesis, solemnly declare that no portion of our thesis has been plagiarized and any material used in the thesis from other sources is properly referenced.

We further understand that if we are found guilty of any form of plagiarism in the thesis work even after graduation, the University reserves the right to revoke our BS degree.

Urooj Afridi Signature: ________________

Alishan Khattak Signature: ________________

____

Verified by Plagiarism Cell Officer
Dated:

Certificate of Approval

The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *Detection of Osteoarthritis using Advance Segmentation in Radiographic Images*, submitted by Urooj Afridi (20P-0192), and Alishan Khattak (21P-8046), in its current form, and it is satisfying the dissertation requirements for the award of Bachelors Degree in Computer Science.

Supervisor

Dr. Muhammad Amin Signature: __________________

____

Coordinator of FYP

FYP Coordinator
National University of Computer and Emerging Sciences, Peshawar

____

Dr. Head of Computer Science Department

HoD of Department of Computer Science
National University of Computer and Emerging Sciences

Acknowledgements

In the name of Allah, the Most Merciful, the Most Beneficent, we begin our Final Year Project (FYP) with heartfelt thanks to Allah, our Creator, for His blessings and guidance. Our sincere appreciation extends to our supervisor, Dr. Muhammad Amin, and all the dedicated teachers whose support has been crucial in navigating this significant project. Each of us has played an important role in moving this project forward. We are all fully committed to making it successful, working hard to bring pride to our parents and teachers. Furthermore, we are indebted to our university for providing the necessary resources and facilities that facilitated the completion of this proposal. The research would not have been possible without their support. We would like to thank all of our friends and family for their understanding, encouragement, and patience throughout this endeavor. Their support has been a constant source of motivation. Lastly, we want to express our gratitude to all those unnamed individuals who have contributed to the development of this project in various ways. Their collective efforts have not gone unnoticed. Thank you for your invaluable contributions.

Urooj Afridi
Alishan Khattak

Abstract

In response to the critical problem of accurately detecting knee osteoarthritis (OA) on X-ray images, our project addresses the limitations associated with previous approaches. These limitations, including data scarcity and varied OA presentations, have greatly hindered accuracy and efficiency in detection. In this study, we propose a comprehensive approach for improving medical image segmentation by leveraging a pre-trained model and innovative architectural modifications. Our methodology involves preprocessing medical images using state-of-the-art techniques, followed by fine-tuning the model to adapt to the specific characteristics of medical imaging data. Our methodology involves employing a specified deep learning architecture, in conjunction with specialized medical imaging datasets specifically curated for this task. We emphasize the use of a specific medical image dataset, which contributes to significantly improved results. Our model's proficiency in segmentation tasks makes it an ideal choice for precisely delineating OA regions within X-ray images. Through meticulous data curation and advanced learning techniques, our model adapts to the unique variations in OA presentations. Our key results will demonstrate that our approach achieves efficient and accurate OA detection and classification, significantly improving clinical diagnosis and patient care efficiency. In conclusion, our project provides a comprehensive response to the challenges of OA detection of the knee, promoting advancements in medical imaging and healthcare.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In our project, we are diving into the world of medical imaging to improve how we detect knee Osteoarthritis(OA) using X-ray images. We know how important it is to spot this condition accurately and quickly. Thus, we have embarked on a journey to develop a specialized system using a carefully selected data set and state-of-the-art image segmentation techniques.

Knee OA is a prevalent and debilitating condition that requires early and accurate diagnosis for effective management. However, conventional approaches to OA detection may be limited by their reliance on manual interpretation or outdated automated methods [1]. Previous research in medical imaging has highlighted the importance of taking advantage of advanced techniques, such as deep learning and image segmentation, to improve the accuracy of OA detection.

Our journey begins with the preprocessing of knee X-ray images, where we ensure each image is perfectly primed for analysis. This involves tasks like resizing to a standard format[2]. In addition, we employ advanced techniques such as data augmentation to diversify our data set and make our model more robust.

With our dataset prepared, we turn to a pretrained model that has already learned a lot from a vast amount of data [3]. We fine-tuned this model to teach it the specific nuances of knee OA detection in X-ray images. Through this process, our model becomes adept at recognizing the subtle signs of OA and precisely delineating affected areas within X-ray scans. This transfer learning strategy forms the basis for our model, facilitating fine-tuning for accurate segmentation of knee OA regions.

Our model, designed for optimal performance in image segmentation, undergoes training to enhance its ability to precisely delineate OA areas within X-ray images. Once our model is fully trained, we put it to work on new knee X-ray images, helping us to identify areas that might have OA. But before we trust its results, we test it a bit to make sure it is accurate and reliable. And, of course, we always keep things ethical and respectful.

In essence, our project is about creating a smart tool that can assist doctors and healthcare professionals in detecting knee OA more efficiently and accurately [4]. By combining advanced image processing techniques with cutting-edge AI models, our objective is to make a meaningful impact on medical imaging and ultimately improve patient care outcomes [5].

Chapter 2

Review of Literature

Our literature review delves into the particulars of methodologies employed in the realm of medical imaging, specifically focusing on knee osteoarthritis (OA) detection using radiographic images. A comprehensive analysis of related studies sheds light on various approaches, including transfer learning with pretrained models such as VGG-16 and ResNet-50 [5]. These studies underscore the importance of meticulous model selection and training for achieving commendable results in disease detection tasks [6]. Additionally, recent advancements in deep learning techniques have shown promising results in automating the detection and diagnosis of knee osteoarthritis, highlighting the potential of convolutional neural networks (CNNs) to extract meaningful features from radiographic images [7]. Notably, paper [7] discusses the utilization of deep learning methods for knee OA diagnosis, emphasizing the significance of early detection to mitigate disease progression.

However, a critical examination reveals certain gaps in the existing body of research, notably in the realms of data handling, ethical considerations, and clinical validation [8]. For instance, while AI-driven models have demonstrated high accuracy in knee OA detection, there is a need for robust validation on diverse patient populations to ensure their efficacy across demographics [9]. Furthermore, ethical considerations regarding patient privacy and data security remain paramount, necessitating clear guidelines and regulations for the responsible deployment of AI in medical imaging [10]. Additionally, challenges related to data scarcity and model generalization highlight the need for further exploration into semi-supervised and self-supervised learning approaches [5], [8].

One notable study contributes significantly to the discourse by presenting an improved model for automating OA severity segmentation [11]. While demonstrating the potential of deep learning in this domain, the study also highlights inherent challenges such as data quality assurance, limitations imposed by single modality focus, and interpretability issues associated with complex models [12]. Similarly, other investigations explore deep learning models for disease classification using medical images, aiming to mitigate challenges arising from data scarcity and generalization issues [13]. Moreover, recent advancements in semi-supervised learning approaches, as demonstrated in papers [12] and [14], have shown promise in leveraging unlabeled data to improve model performance, thereby addressing the challenge of limited annotated medical image datasets.

Furthermore, an exploration into medical image segmentation reveals the intricacies associated with data availability and model transferability [15]. Notably, efforts to segment effusions in knee X-ray datasets underscore the crucial role of enhanced pretraining strategies and the importance of diverse datasets in advancing segmentation methodologies [13]. The application of transformer-based models, as discussed in paper [16], shows promise in improving medical image segmentation tasks, particularly in handling large-scale datasets and capturing long-range dependencies within images. Additionally, architectures like U-Net have emerged as frontrunners in segmentation tasks due to their inherent capability to capture intricate spatial features [16].

However, despite the progress made in the field, there remains a gap in the literature concerning the integration of multiple models within a unified framework to optimize segmentation processes further. Our project aims to bridge this gap by adopting a holistic approach that not only leverages pretrained models for initial training but also integrates another model to refine results within a smaller yet representative dataset [17]. This innovative strategy holds promise in enhancing the precision of knee osteoarthritis detection and segmentation. Furthermore, recent advancements in self-supervised learning techniques, as discussed in paper [16], offer a promising avenue for addressing the challenge of limited annotated data by leveraging unlabeled medical imaging datasets for pretraining purposes.

In summary, our literature review serves as a cornerstone for understanding existing methodologies, their limitations, and the avenues for future research. By addressing key challenges such as data scarcity, model robustness, and clinical validation, we endeavor to propel the field of medical imaging forward and contribute to improved patient care outcomes. Additionally, the integration of ethical considerations and patient-centric approaches will be pivotal in ensuring the responsible and effective deployment of AI-driven solutions in medical imaging [11], [12].

Chapter 3

Project Vision

3.1 Problem Statement

The problem we aim to address is the need for accurate and efficient detection of knee osteoarthritis (OA) using X-ray images. Despite advances in medical imaging and deep learning techniques, current approaches often face challenges such as data scarcity, model robustness, and generalization issues. Our project seeks to overcome these challenges by developing a specialized system that can reliably identify and delineate OA regions within knee X-ray images, thereby facilitating early diagnosis and effective treatment planning for patients with knee OA. Through the integration of advanced preprocessing techniques, transfer learning with pretrained models, and innovative architectural modifications, we strive to enhance the accuracy and efficiency of knee OA detection, ultimately improving clinical decision-making and patient care outcomes in the field of medical imaging and healthcare.

3.2 Business Opportunity

The development of an accurate knee osteoarthritis (OA) detection system using X-ray images presents a promising business opportunity in the healthcare sector. By providing innovative medical technology solutions, such as AI-driven diagnostic tool, healthcare providers can enhance patient care and attract more patients, thereby differentiating themselves in the market. Moreover, there is potential in medical device manufacturing to develop specialized imaging equipment with advanced analytics for real-time insights. Consulting services in data analytics and AI can also be used to help healthcare organizations implement and optimize AI-driven solutions.

3.3 Objectives

The objectives of our project are to enhance knee osteoarthritis (OA) detection accuracy using X-ray images, ensuring robust adaptability and reducing annotation burden. We aim to validate clinical effectiveness, advance medical image analysis, and address data scarcity limitations, ultimately improving patient care.

3.4 Project Scope

The project scope encompasses precise osteoarthritis (OA) in knee joints, with the aim of improving the detection time and accuracy through advanced methods. In addition, it focuses on improving model generalization through data enhancement and enabling rapid adaptation to new medical cases with specialized techniques.

3.5 Constraints

Constraints: The development of the knee osteoarthritis detection system will be limited by factors such as limited annotated data, available resources, including time, budget, and computational capabilities. Furthermore, limitations in hardware and software frameworks must be considered to ensure the efficient deployment of the system in various medical imaging settings.

3.6 Stakeholders Description

Our project involves key stakeholders such as healthcare providers (hospitals, clinics), patients, medical device manufacturers, and AI technology developers. Healthcare providers benefit from improved diagnostic accuracy and streamlined patient care processes, while patients receive timely and accurate diagnoses. Medical device manufacturers focus on developing innovative imaging solutions, and AI technology developers create robust deep learning models for ethical use in healthcare.

3.6.1 Stakeholders Summary

Our stakeholders include healthcare providers seeking improved diagnostics, patients in need of timely diagnoses, medical device manufacturers developing innovative solutions, and AI technology developers ensuring ethical use of AI.

3.6.2 Key High Level Goals and Problems of Stakeholders

Key High Level Goals and Problems of Stakeholders: Healthcare providers aim to improve diagnostic accuracy, improve patient outcomes, and optimize resource utilization. Patients seek timely and accurate diagnoses, access to effective treatment, and assurance of privacy and confidentiality regarding their medical data. Medical device manufacturers aspire to develop cutting-edge imaging technologies, address regulatory requirements, and meet market demand. Developers of AI technologies strive to create reliable and interpretable AI models, address data privacy concerns, and ensure ethical use of AI in healthcare applications.

Chapter 4

Software Requirements Specifications

This chapter includes the functional and non-functional requirements of the project.

4.1 List of Features

1. Knee Osteoarthritis Detection: Our system provides a reliable and accurate method for detecting knee osteoarthritis in X-ray images.
2. Automated Segmentation: We implement automated segmentation to precisely delineate the regions of osteoarthritis within X-ray images.
3. Transfer Learning: We are incorporating transfer learning techniques to efficiently handle data scarcity and recognize diverse presentations of osteoarthritis.
4. Specialized Medical Image Dataset: we are using specific medical image datasets curated for the detection of knee osteoarthritis, improving results compared to using generic or public datasets.
5. Model Architecture: We employ an efficient deep learning architecture, fine-tuned through transfer learning, optimized for knee X-ray images.
6. Data Curation: Our system addresses data bias through meticulous data curation, ensuring the reliability and quality of the dataset.
7. Innovative OA Detection Approach: introduces an innovative osteoarthritis detection method, focusing on feature extraction and segmentation capabilities.

8. Validation: We rigorously validate the accuracy and precision of the model to ensure reliable results.
9. Ethical Considerations: We are addressing ethical considerations surrounding patient data, data security, and potential biases in the model predictions.
10. Clinical Application: Our aim is to improve auxiliary clinical diagnosis and patient diversion in healthcare settings.
11. Future proofing: We are designing our system to accommodate future advancements in medical imaging and technology.

These are some of the features that outline the core functionalities and goals of our project, ensuring that our system meets the needs of accurate osteoarthritis detection and classification in the context of medical image analysis.

4.2 Functional Requirements

1. Image Input: The system will accept knee X-ray images in standard digital formats as input.
2. Automated Segmentation: The system will provide automated segmentation of knee osteoarthritis regions in the input X-ray images.
4. Dataset Management: The system will allow the import, curation and management of specialized medical image datasets for model training.
5. Validation: The system will include validation tools to assess the accuracy and precision of the model results.
6. User Interface: The system will provide an intuitive user interface for users to interact with the software.

4.3 Quality Attributes

The system aims to achieve accurate detection, improved efficiency, and a more robust model while maintaining simplicity and utilizing smaller data sets.

4.4 Non-Functional Requirements

1. Performance: Our system will deliver segmentation results within a reasonable time frame, ensuring real-time or near-real-time processing.
2. Accuracy: The system will achieve a high level of accuracy in detecting osteoarthritis regions in X-ray images.
3. Scalability: The system will be scalable to accommodate a growing database of medical images and potential future updates.
4. Data Security: The system will implement robust data security measures to protect patient information and adhere to healthcare privacy regulations.
5. Ethical Considerations: The system will be designed to minimize potential biases and ensure ethical use of AI in healthcare.
6. User training: The system will provide user training and documentation to ensure that healthcare professionals can use the software effectively.
7. Compatibility: The system will be compatible with standard operating systems and hardware commonly used in healthcare settings.
8. Reliability: The system will be reliable, with minimal downtime and a backup and recovery system in place.
9. Adaptability: The system will adapt to variations in data quality, patient populations, and imaging equipment to ensure robust performance.
10. Future proofing: The system will be designed to accommodate future advancements in medical imaging and AI technology.

These functional and non-functional requirements provide a comprehensive framework for our project, ensuring that it meets the needs of healthcare professionals and adheres to industry standards and ethical considerations.

4.5 Use Case Diagram

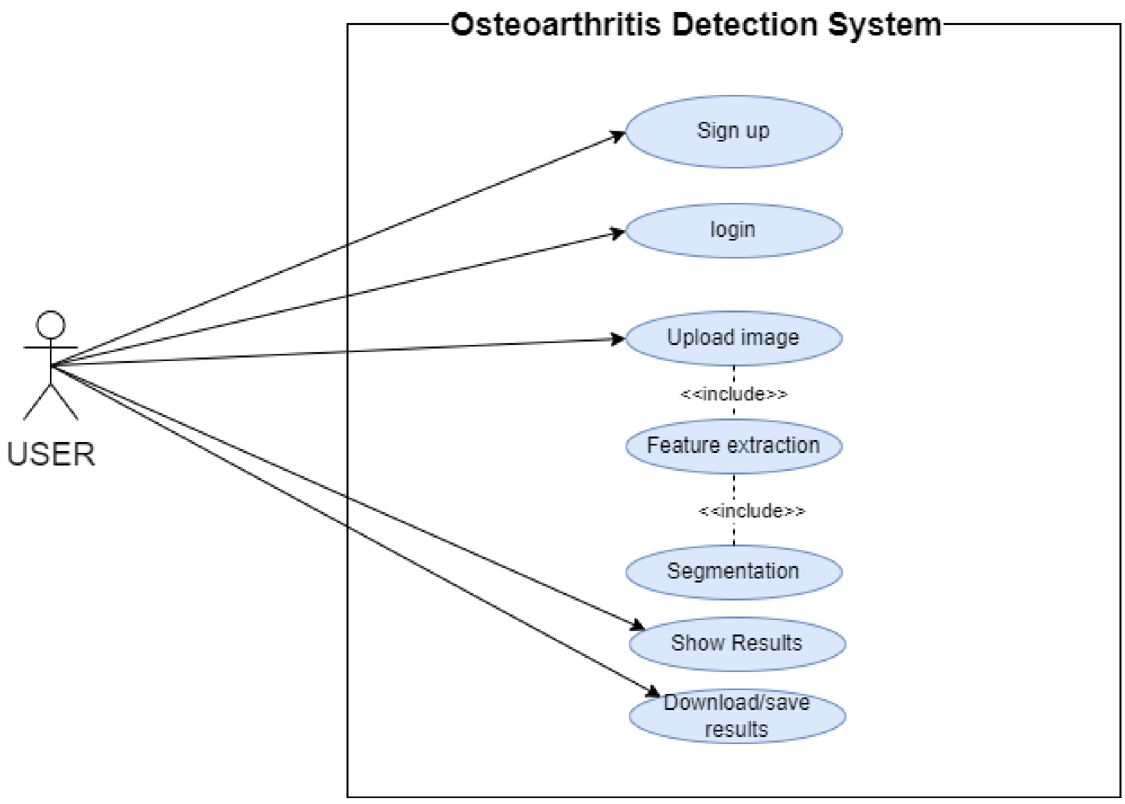


Figure 4.1: Use Case Diagram

4.6 Sequence Diagrams/System Sequence Diagram

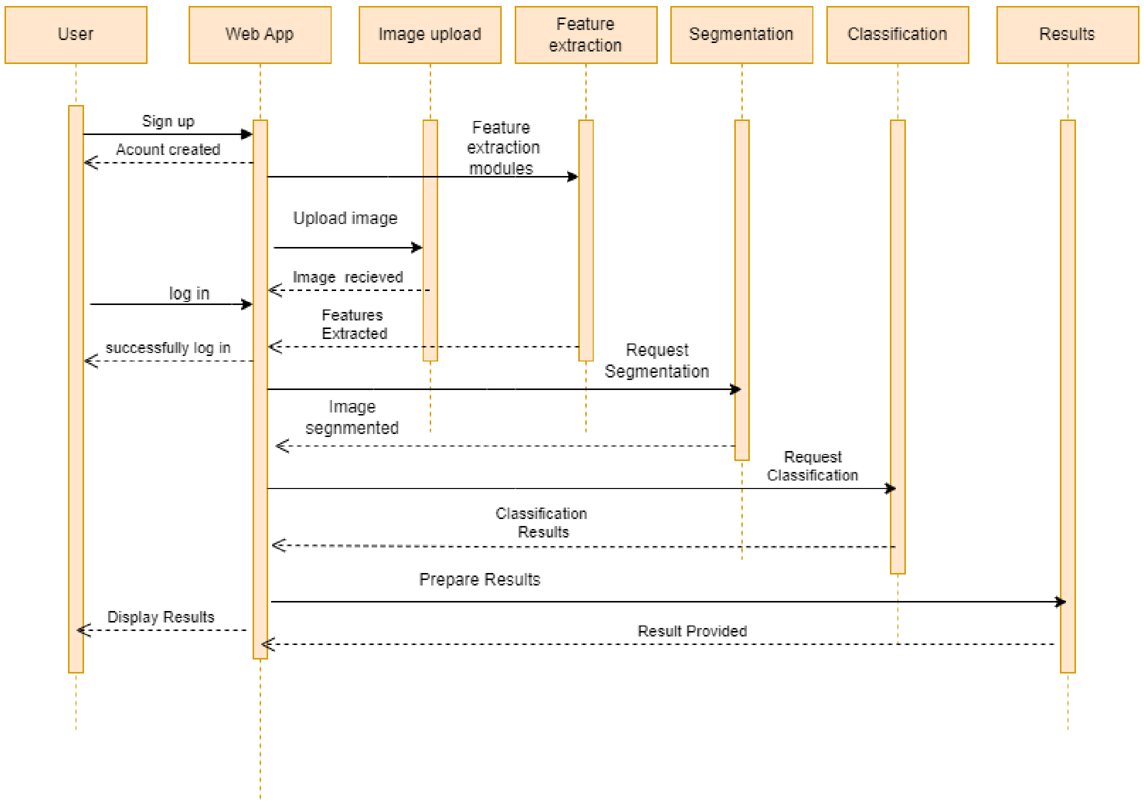


Figure 4.2: System Sequence Diagram

4.7 Test Plan (Test Level, Testing Techniques)

Component Testing:

Assess the functionality and performance of individual components within our knee osteoarthritis detection system, including preprocessing modules, segmentation algorithms, and user interface elements.

System-wide Testing:

Testing the entire system as a unified entity to verify its compliance with predefined specifications and requirements.

Efficiency Evaluation:

Analyze the effectiveness and computational efficiency of the system's algorithms and processes, aiming to optimize resource utilization and overall performance.
Validation Testing:
Validating the system performance using real-world knee X-ray images to ensure that it meets the expectations of stakeholders, including healthcare providers and patients.

4.8 Software Development Plan

Our software development plan outlines the entire software development lifecycle, including requirements collection, design, implementation, testing, deployment, and maintenance. It includes a detailed breakdown of tasks, timelines, roles, and responsibilities, tools, and technologies used, as well as risk management strategies to ensure successful project completion.

Chapter 5

Iteration Plan

Our project journey unfolds in several distinct phases, each marked by significant milestones and progress towards our ultimate goals.

Proposal Defense: We commenced by presenting our project idea, encompassing its introduction, motivation, scope, problem statement, proposed solution, and an initial literature review.

Mid-Term Presentation FYP 1: We delved into an exhaustive examination of the relevant literature, extracting insights to inform our project development trajectory. Visual representations were crafted to elucidate the architecture and key components or the system diagrams of our proposed solution along with these we meticulously outlined the functional and non-functional requirements essential for our system's success.

Final Presentation FYP 1: Data Collection and Preprocessing was performed by gathering pertinent data ensued, followed by rigorous preprocessing employing techniques such as resizing and data augmentation.

First Presentation FYP 2: In our initial presentation for the FYP 2 phase, we showcased significant progress in key aspects of our project. We embarked on the implementation of our model, leveraging a pre-trained architecture to kickstart the process. While the initial pre-training laid the foundation, our focus transitioned towards fine-tuning the model to ensure alignment with the unique characteristics of our dataset. In addition, strides were made in the development of our web application, with an emphasis placed on front-end design and user interface development. As we presented our progress, it became evident that we were well-positioned to advance towards the final stages of our project, poised to achieve our objectives with diligence and precision.

Final Presentation FYP 2: As we approached the second presentation for the Final FYP 2 phase, building upon the groundwork laid in the initial presentation, we have finalized the crucial elements of our project. Simultaneously, our focus is on comprehensive fine-tuning of our model, aimed at enhancing its performance and accuracy across diverse datasets and scenarios. Moreover, significant strides are anticipated in the development of our web application, with plans in place for the seamless integration of both front-end and back-end functionalities. With these pivotal tasks, our project embodies the spirit of ongoing exploration and innovation, driven by a steadfast commitment to achieving excellence in medical imaging and healthcare.

Each stage unfolded seamlessly, guided by our commitment to excellence and propelled by a shared vision of impactful innovation in the domain of medical imaging and healthcare

Chapter 6

Iteration 1

The initial stage of our project is projected to end before the midpoint of FYP-1. During the mid-term FYP1 phase, our efforts were focused on deep-dive into the intricate domain of osteoarthritis segmentation. This phase involved extensive research and the refinement of techniques tailored for segmentation and detection within this challenging problem domain. Our research journey during this stage consisted of the following crucial steps.

In-depth review of the literature: A comprehensive examination of a substantial collection of 40 research papers provided invaluable insight into methods for automated osteoarthritis detection and segmentation. Dataset Exploration: The exploration of high-quality datasets relevant to the problem domain was carried out to ensure the applicability and relevance of our research. Domain Knowledge Exploration: A meticulous exploration of the extensive knowledge within the domain focused on extracting essential insights, leading to the precise articulation of a clearly defined problem statement.

6.1 Introduction:

Our project is dedicated to enhancing the accuracy and efficiency of knee Osteoarthritis (OA) detection through the utilization of advanced medical imaging techniques, particularly focusing on X-ray images. Rapid and accurate detection of knee OA is essential for effective patient management, but current approaches often face significant challenges, including data scarcity and variability in the presentations of OA. To address these limitations, our research endeavors to develop a comprehensive approach that takes advantage of state-of-the-art image segmentation techniques and deep learning models. Through meticulous data curation, innovative architectural modifications, and fine-tuning of pre-trained models, our aim is to develop a robust system capable of precisely delineating OA regions within knee X-ray images.

6.2 Related Work:

The foundation of our research lies in a critical review of the existing literature, which has provided invaluable information on the methodologies used in medical imaging for the detection of knee OA using radiographic images. Through an extensive examination of 32 research papers, we have gained extensive knowledge on automated osteoarthritis detection and segmentation methods [18], [19], [20]. These studies underscore the importance of leveraging advanced machine learning techniques, such as convolutional neural networks (CNNs), and architectures like VGG-16 and ResNet-50, which have shown promising results in extracting meaningful features from radiographic images [21], [12], [22].

Additionally, our exploration has extended to identifying and exploring high-quality datasets pertinent to the problem domain, ensuring the relevance and applicability of our research [23], [24], [25]. Incorporation of MRI methodologies has also been pivotal, with studies highlighting the superiority of MRI in assessing knee structures and helping in early OA diagnosis and prognosis through models based on machine learning and deep learning [26]. Furthermore, the integration of explainable artificial intelligence (XAI) techniques has been explored to improve the interpretability and trustworthiness of AI models in clinical practice, addressing the critical need for transparency in AI-driven decision-making processes [27], [28].

Our thorough investigation of related work serves as the cornerstone of our project, bridging the gap between existing methodologies and the evolving needs of accurate knee OA detection in clinical practice. Recent advances in segmentation tools for musculoskeletal structures and the development of predictive models that incorporate MRI markers underscore the importance of a multidisciplinary approach to improve diagnosis and patient outcomes [25], [29], [30]. By addressing key challenges such as data scarcity, model robustness, and clinical validation, our research efforts aim to advance the state-of-the-art in medical imaging and contribute meaningfully to the improvement of patient care outcomes [31], [32].

6.3 Problem Statement:

Our project aims to tackle the critical challenge of accurately detecting kosteoarthritisitis (OA) using X-ray images. Despite advances in medical imaging and deep learning, current approaches face issues with data scarcity, model robustness, and generalization. To address these challenges, we are developing a specialized system to reliably identify and delineate OA regions within knee X-ray images. Through advanced preprocessing techniques, transfer learning, and architectural modifications, our goal is to improve detection accuracy and efficiency, ultimately improving patient care outcomes in medical imaging and healthcare.

6.4 Data Representation

The dataset [33] we used initially consists of 1650 digital X-ray images of knee joints collected from well-reputed hospitals and diagnostic centers. X-ray images were acquired using the PROTEC PRS 500E X-ray machine, known for its high-resolution imaging capabilities. These original images are 8-bit grayscale, ensuring sufficient detail for accurate analysis and diagnosis. Each knee radiographic image has been meticulously annotated and labeled according to Kellgren and Lawrence grades by two experienced medical experts, ensuring high reliability and consistency in the labeling process.

We are working with five classes within the dataset, which include normal, doubtful, mild, moderate, and severe cases of knee osteoarthritis. This classification provides a comprehensive range for assessing the progression of the disease, facilitating the development and validation of machine learning models for the detection and classification of knee OA [33].

Given that the initial dataset was relatively small, we augmented it by incorporating additional datasets and applying various preprocessing steps. These steps included image normalization, enhancement, and enhancement techniques, which increased the size and variability of the data set, thus improving its suitability for training robust machine learning models.

Below is the visual representation of initial dataset:

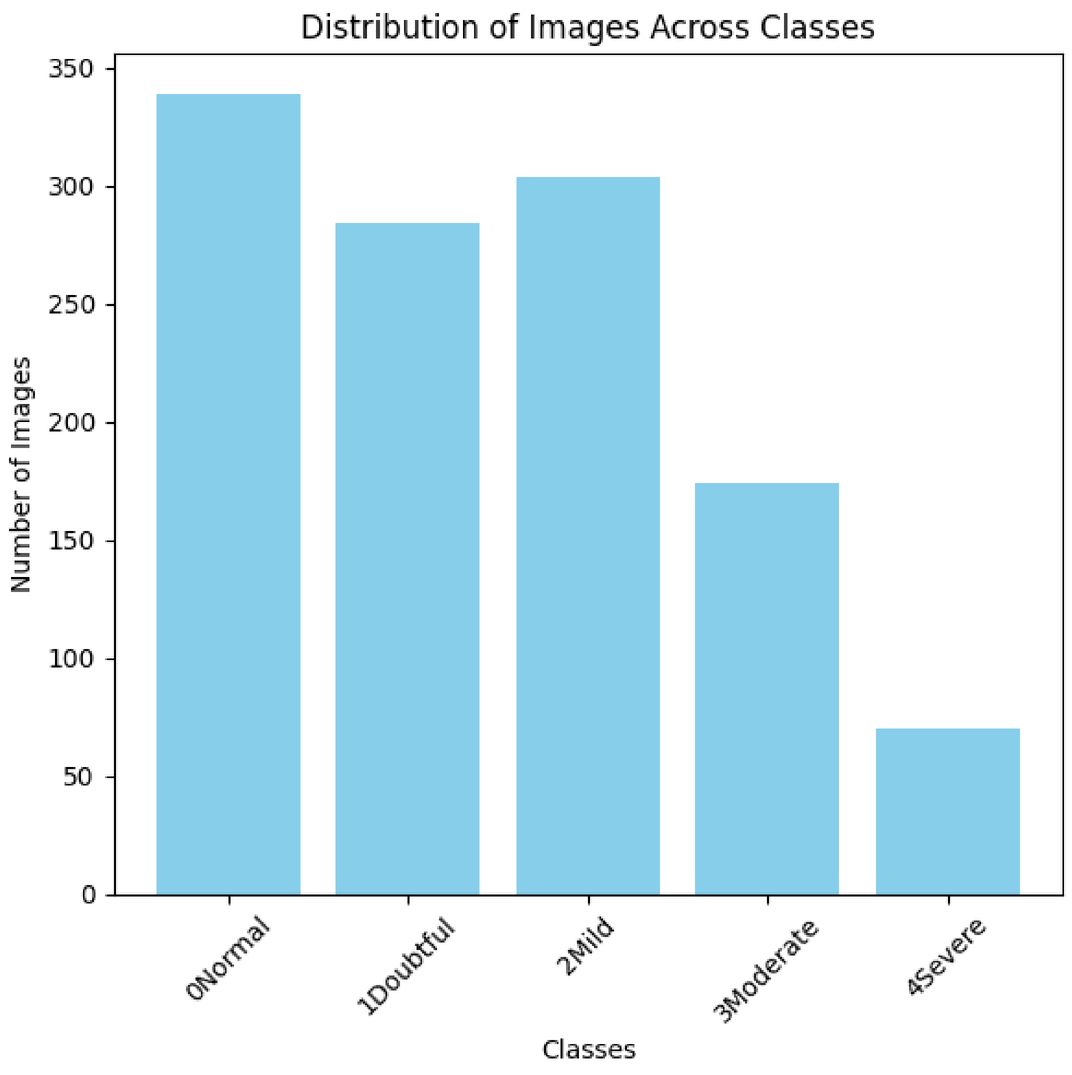


Figure 6.1: Initial Dataset Statistics

6.5 Activity Diagram

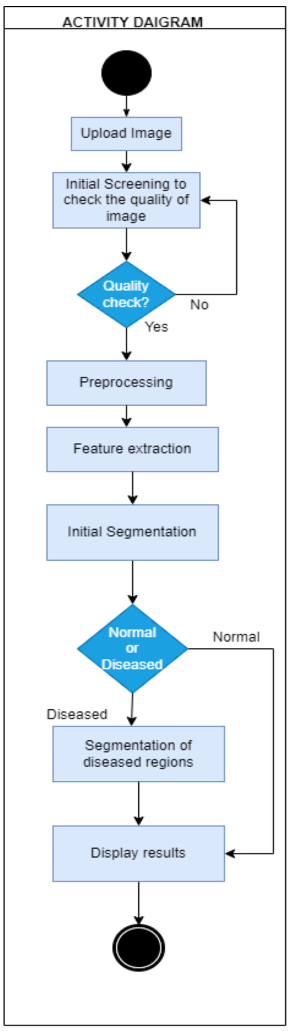


Figure 6.2: Activity Diagram

Chapter 7

Iteration 2

This iteration, completed at the end of FYP-1, marks a significant milestone in our project. We have successfully completed the system design phase, meticulously crafting an architecture that encompasses the complexities of detecting knee osteoarthritis from X-ray images. Our design incorporates a robust methodology for data collection, ensuring that we have a diverse and representative set of data for training and evaluation purposes. After collecting the dataset, we applied various preprocessing techniques, as the initial data set was small, to enhance its size and variability.

With the completion of these key components, we moved forward with the implementation phase, where we brought our design to life and conducted rigorous testing and validation to ensure its effectiveness.

7.1 Proposed System Diagram

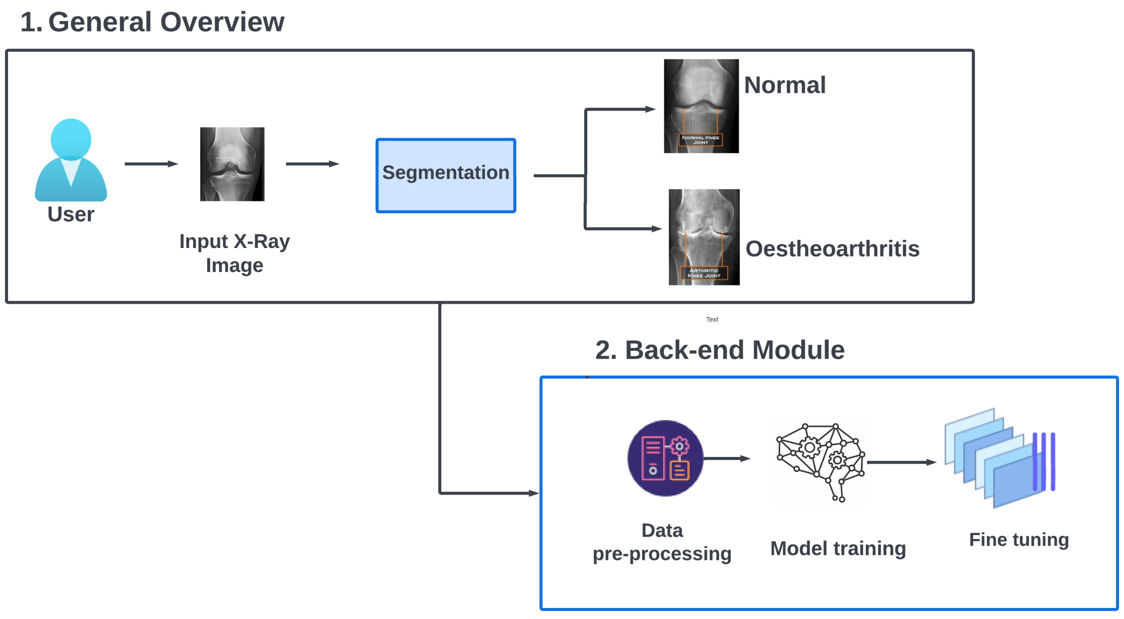


Figure 7.1: Proposed System Diagram

7.2 Architecture Diagram

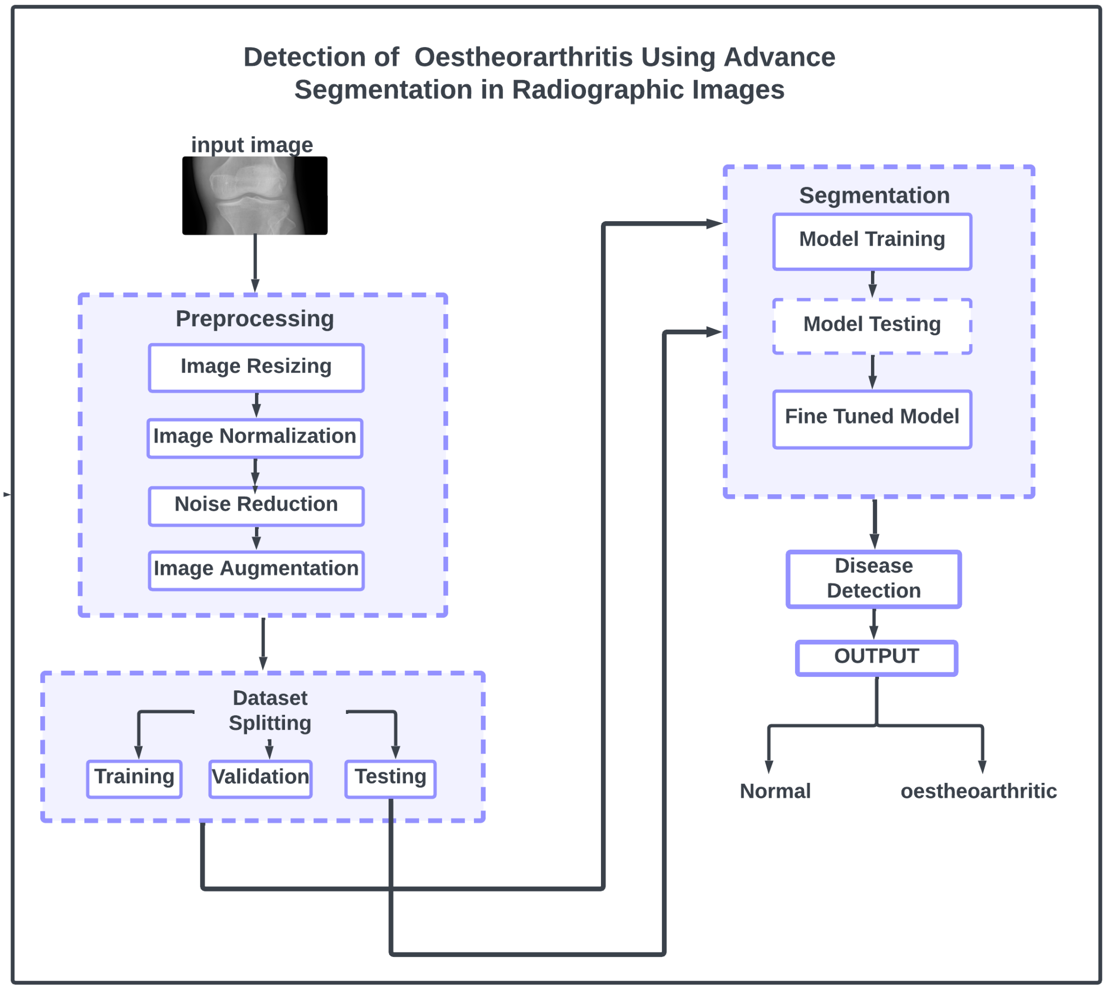


Figure 7.2: Architecture Diagram

7.3 Data Preprocessing

Data preprocessing is a crucial step in the development of any machine learning model, particularly in medical imaging where the quality and consistency of the data significantly impact the model's performance. In our project on knee osteoarthritis detection, we performed several pre-processing techniques to ensure the suitability of the data set for training robust models. These techniques include resizing and augmentation. Each of these methods is detailed below:

7.3.1 Resizing

Resizing is the process of altering the dimensions of the images to a uniform size. This step is essential because it standardizes the input size for the neural network, thus reducing computational complexity and memory usage during training.

- Method: All X-ray images were resized to a fixed dimension of 256x256 pixels.
- Purpose: Ensures consistency across the dataset and compatibility with the input layer of the neural network.

```python

def process_image(image_path, output_path):
    image = Image.open(image_path)
    w, h = image.size

    # Calculating the aspect ratio
    aspect_ratio = w / h

    # Determining the new size keeping the aspect ratio
    if aspect_ratio > 1:  # Width is greater than height
        new_w = 256
        new_h = int(new_w / aspect_ratio)
    else:  # Height is greater than or equal to width
        new_h = 256
        new_w = int(new_h * aspect_ratio)

    # Resizing the image
    resized_image = image.resize((new_w, new_h), Image.Resampling.
   LANCZOS)

    # Calculating padding
    pad_h = (256 - new_h) // 2
    pad_w = (256 - new_w) // 2

    # Padding the resized image to make it 256x256
    padded_image = ImageOps.expand(resized_image, (pad_w, pad_h, 256 - new_w - pad_w, 256 - new_h - pad_h), fill=0)

    padded_image.save(output_path)

    return image, padded_image

def process_folder(input_folder, output_folder):
    original_images = []
    resized_images = []

    # Iterating through the directory
    for root, dirs, files in os.walk(input_folder):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):
                input_path = os.path.join(root, file)
                # Creating the corresponding output path
                relative_path = os.path.relpath(input_path, input_folder)
                output_path = os.path.join(output_folder, relative_path)

                # Creating the output directory if it doesn't exist
                os.makedirs(os.path.dirname(output_path), exist_ok=True)

                # Processing the image
                original, resized = process_image(input_path, output_path)
                original_images.append((file, original))
                resized_images.append((file, resized))

    # Show the images
    show_images(original_images, resized_images)

def show_images(original_images, resized_images):
    # Display the first two pairs of images
    for i in range(2):
        print(f"Displaying Original Image: {original_images[i][0]}"
    )
        original_images[i][1].show(title=f"Original: {
    original_images[i][0]}")

        print(f"Displaying Resized Image: {resized_images[i][0]}")
        resized_images[i][1].show(title=f"Resized: {resized_images[
    i][0]}")

input_folder = 'splitted_new_dataset'
output_folder = 'dfgjdfgdfgdfgdf'

# Processing the entire folder
process_folder(input_folder, output_folder)
```

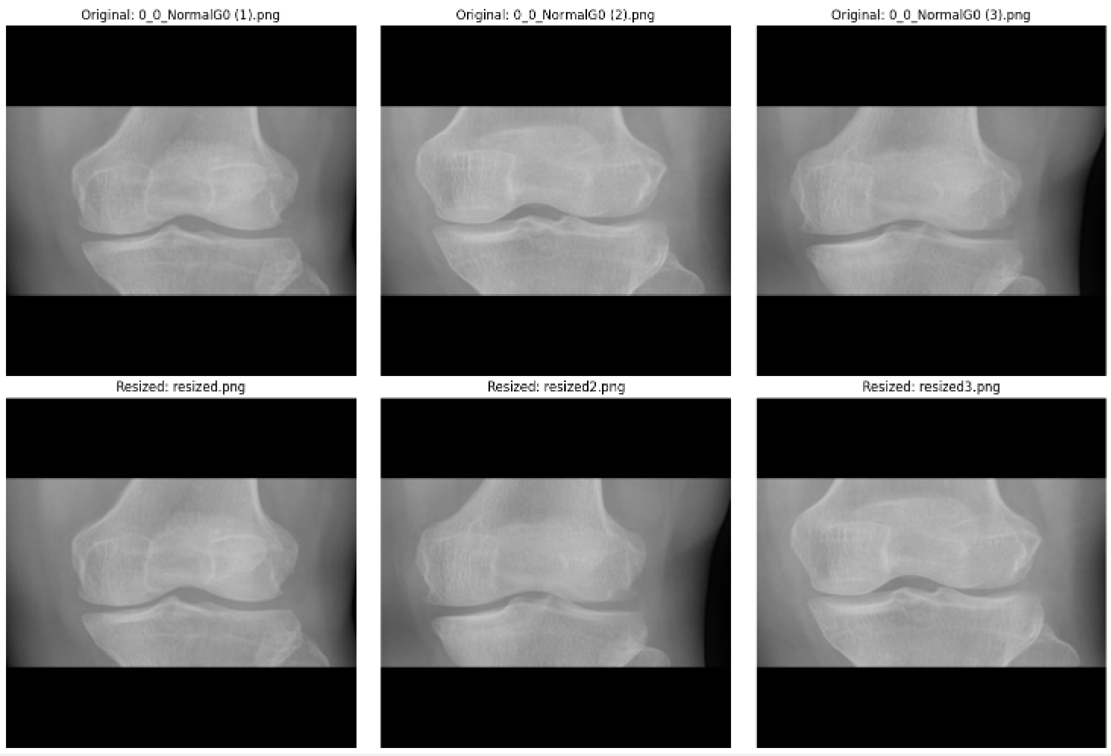

Original: 0_0_NormalG0 (1).png
Original: 0_0_NormalG0 (2).png
Original: 0_0_NormalG0 (3).png
Resized: resized.png
Resized: resized2.png
Resized: resized3.png

Figure 7.3: Sample Resized images

7.3.2 Convertion to RGB

Conversion to RGB is the process of standardizing the color format of all images in the dataset to ensure uniformity. This step is crucial as it helps in maintaining consistency across the dataset, which is essential for accurate model training.

- Method: All images in the dataset were checked for their color modes. Any image that was not in RGB mode was converted to RGB.
- Purpose: Ensures that all images are in the same color mode, preventing discrepancies during model training and ensuring compatibility with the neural network's input layer.

```
def print_image_sizes_and_modes(directory):
    counter = Counter()
    mode_counter = Counter()
    for root, dirs, files in os.walk(directory):
        for filename in files:
            if filename.endswith(".jpg") or filename.endswith(".png
"):
                img = Image.open(os.path.join(root, filename))
                size = img.size
                mode = img.mode
                print(f"The size of {filename} is {size} and mode
is {mode}")
                counter[size] += 1
                mode_counter[mode] += 1
    print("Sizes and their counts:")
    total = 0
    for size, count in counter.items():
        print(f"Size: {size}, Count: {count}")
        total += count
    print(f"Total count of all sizes: {total}")
    print("Modes and their counts:")
    for mode, count in mode_counter.items():
        print(f"Mode: {mode}, Count: {count}")

print_image_sizes_and_modes('splitted_new_dataset')


def convert_images_to_rgb(directory):
    for root, dirs, files in os.walk(directory):
        for filename in files:
            if filename.endswith(".jpg") or filename.endswith(".png
"):
                img_path = os.path.join(root, filename)
                with Image.open(img_path) as img:
                    if img.mode != 'RGB':
                        rgb_img = img.convert('RGB')
                        rgb_img.save(img_path)
                        print(f"Converted {filename} to RGB mode")

convert_images_to_rgb('splitted_new_dataset')
```

7.3.3 Data Augmentation

Data augmentation is a technique used to artificially increase the size of the data set by creating modified versions of the original images. This step helps to improve the robustness and generalization ability of the model.Various augmentation techniques were applied for this purpose.

```python

# Defining parameters
target_count = 1500  # Target count for each class
img_size = (256, 256)  # Size of the images for augmentation

# Function to perform augmentation
def augment_images(images, output_dir, target_count):
    print(f"Augmenting images for class {output_dir.split('/')[-1]}
   ")
    datagen = ImageDataGenerator(
        rotation_range=20,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode="nearest"
    )

    # Ensure the output directory exists
    os.makedirs(output_dir, exist_ok=True)

    # Performing augmentation until the target count is reached
    count = len(images)
    while count < target_count:
        for image in images:
            img = cv2.imread(image)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = img.reshape((1,) + img.shape)
            for batch in datagen.flow(img, batch_size=1, save_to_dir=output_dir, save_prefix='aug', save_format='jpg'):
                count += 1
                print(f"Image created: {count}/{target_count}", end='\r')
                if count >= target_count:
                    break
            if count >= target_count:
                break

# Iterating through class directories
data_dir = "splitted_new_dataset/train"
for class_name in os.listdir(data_dir):
    class_dir = os.path.join(data_dir, class_name)
    if os.path.isdir(class_dir):
        images = [os.path.join(class_dir, f) for f in os.listdir(class_dir) if f.endswith(".png")]
        count = len(images)
        print(f"Class: {class_name}, Count before augmentation: {count}")

        # Balancing class count
        if count > target_count:
            np.random.shuffle(images)
            images = images[:target_count]
        elif count < target_count:
            augment_images(images, class_dir, target_count)

        images = [os.path.join(class_dir, f) for f in os.listdir(class_dir) if f.endswith(".png")]
        count = len(images)
        print(f"Class: {class_name}, Count after augmentation: {count}")

print("All classes processed.")
```

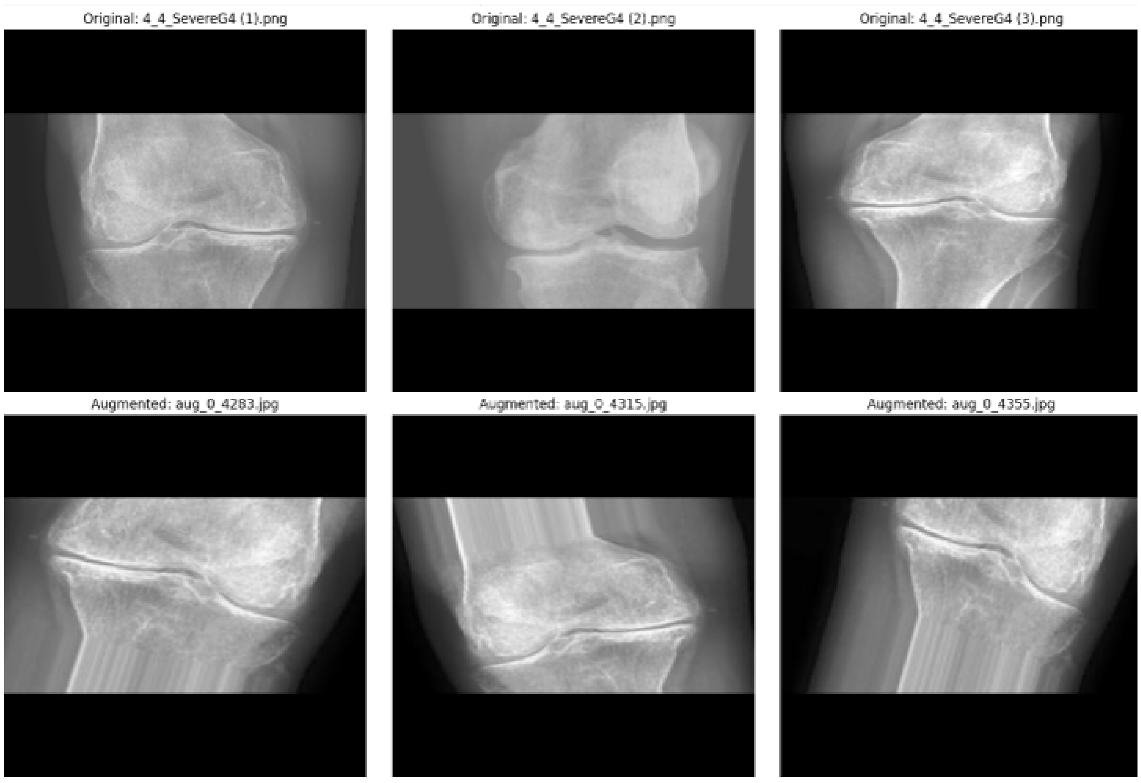

Original: 4_4_SevereG4 (1).png
Original: 4_4_SevereG4 (2).png
Original: 4_4_SevereG4 (3).png
Augmented: aug_0_4283.jpg
Augmented: aug_0_4315.jpg
Augmented: aug_0_4355.jpg

Figure 7.4: Sample Augmented images

Chapter 8

Iteration 3

Initially we made significant progress in laying the groundwork for our system's structural design. Now this phase encompasses the development of the system architecture and the implementation of core functionalities.

One of the primary components of our system architecture is the integration of a deep learning framework for the detection of knee osteoarthritis from X-ray images. Using existing frameworks such as TensorFlow or PyTorch, we have initiated the construction of neural network models tailored to our specific task. These models are designed to process X-ray images efficiently and accurately identify regions indicative of osteoarthritis.

Additionally, we have established a data pipeline for dataset management, preprocessing, and augmentation. Through careful curation and augmentation techniques, we aim to enhance the robustness and generalization capabilities of our models.

By summarizing the previous steps, we prepared our dataset and ensured the robustness of our model. Initially, we collected a dataset consisting of digital X-ray images of the knee joint from reputable hospitals and diagnostic centers. These images, acquired using the PROTEC PRS 500E X-ray machine, were annotated by two medical experts according to the Kellgren and Lawrence grading system. To address the relatively small size of our initial dataset, we incorporated an additional dataset and applied a range of preprocessing techniques such as resizing and augmentation. This thorough preparation has laid a solid foundation for the subsequent phases of model training and evaluation.

8.1 Dataset Splitting

Splitting the dataset into training, testing, and validation sets is a crucial step to ensure that the model can generalize well to new, unseen data. In our project, the dataset is divided into three parts: 80% for training, 10% for testing, and 10% for validation. This distribution allows the model to learn from the majority of the data while being evaluated and fine-tuned on separate subsets.

8.1.1 Training Count

- Class: 0 Normal, Count: 2714

- Class: 1 Doubtful, Count: 2275

- Class: 2 Mild, Count: 2433

- Class: 3 Moderate, Count: 1499

- Class: 4 Severe, Count: 1455

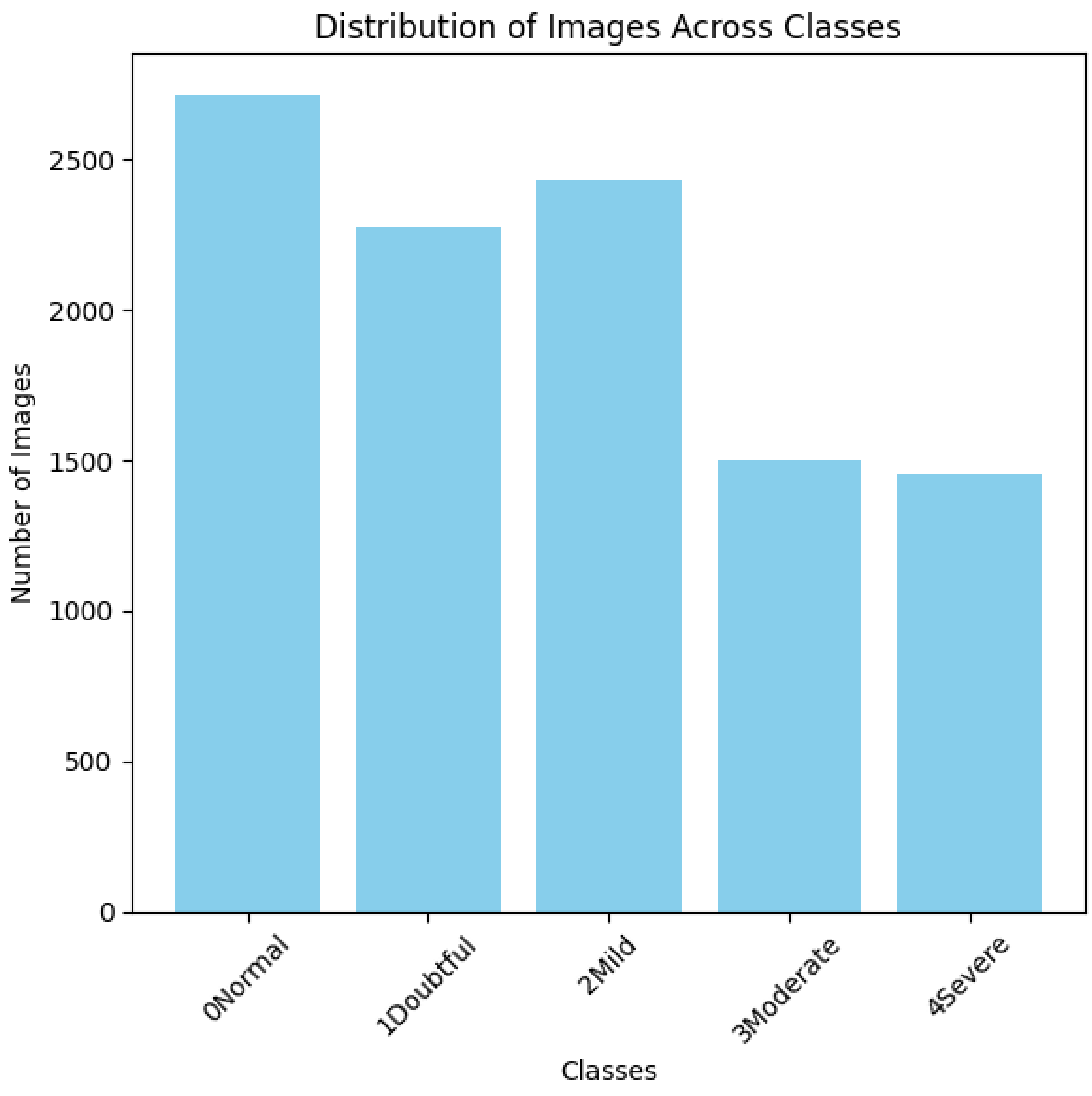


Figure 8.1: Taining Dataset

8.1.2 Testing Count

- Class: 0 Normal, Count: 340
- Class: 1 Doubtful, Count: 285
- Class: 2 Mild, Count: 305
- Class: 3 Moderate, Count: 174

- Class: 4 Severe, Count: 72

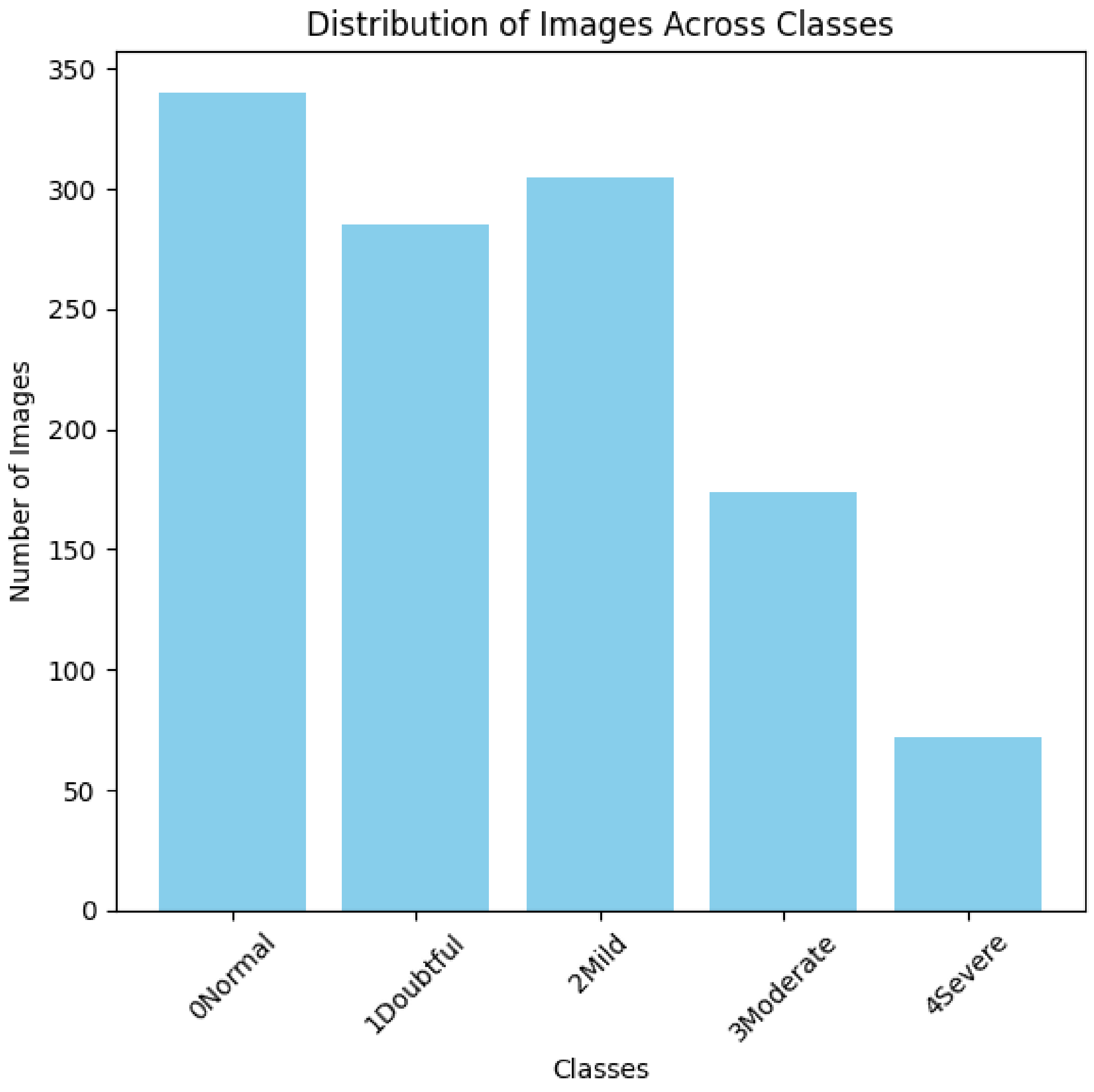


Figure 8.2: Testing dataset

8.1.3 Validation Count

- Class: 0 Normal, Count: 339
- Class: 1 Doubtful, Count: 284
- Class: 2 Mild, Count: 304

- Class: 3 Moderate, Count: 174
- Class: 4 Severe, Count: 70

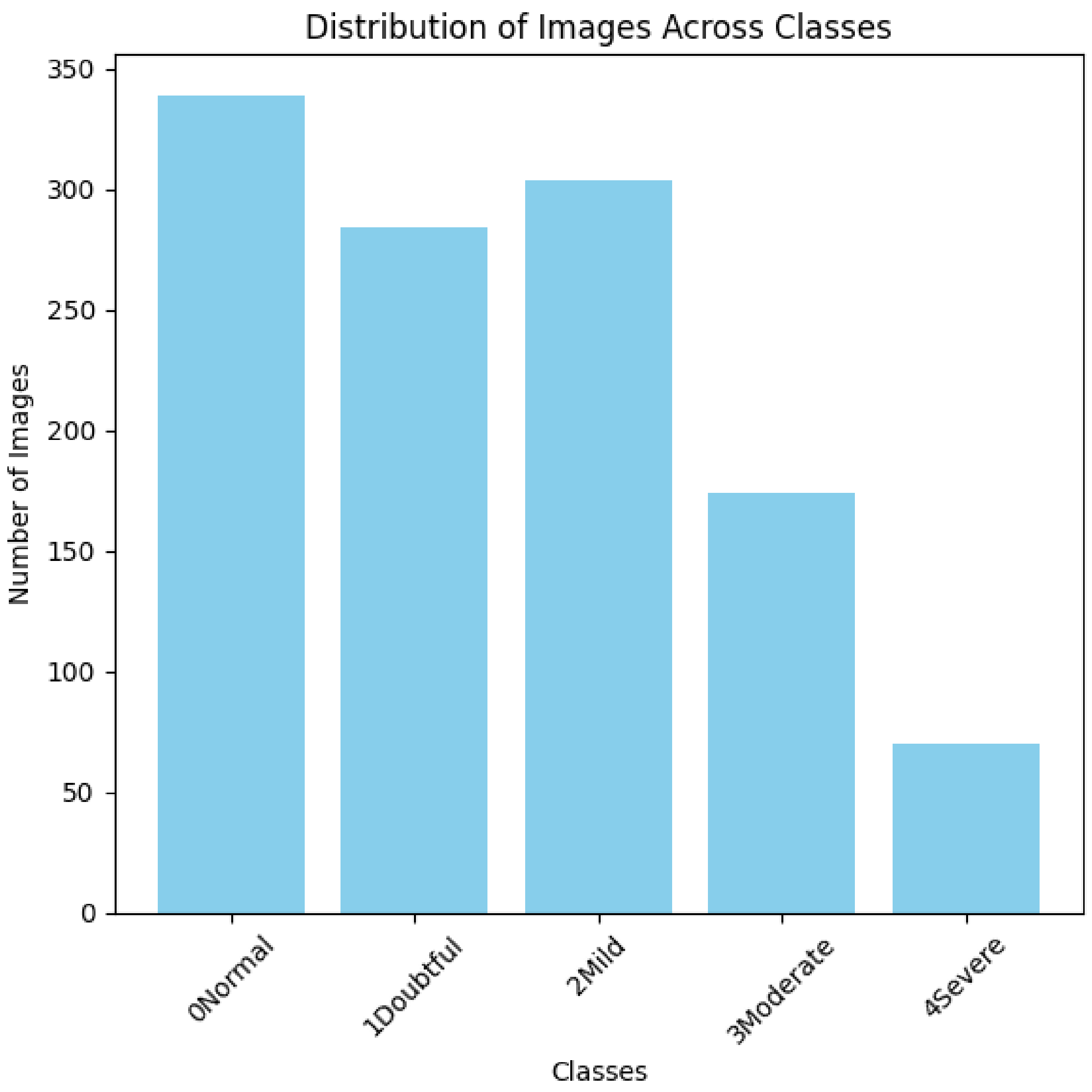


Figure 8.3: Validation Dataset

This systematic division ensures that each class is proportionally represented across all subsets, providing a balanced distribution for training, testing, and validation phases.

8.2 Model Implementation

In the previous steps, we completed data preprocessing and splitting, which laid the groundwork for training our model. With the dataset ready, we explored various models to identify the most effective approach for our task of detecting osteoarthritis (OA) from X-ray images.

8.2.1 Initial Attempts with Pretrained Models

8.2.1.1 UNet with Pretrained ResNet50

We first experimented with a UNet model incorporating a pretrained ResNet50 as its backbone. This combination was chosen for its proven capabilities in image segmentation tasks. Despite its potential, the model only achieved 46% accuracy. This result indicated that the architecture did not adequately capture the features necessary for effective OA detection.

8.2.1.2 UNet with Pretrained ResNet50V2

Next, we tried a UNet model with a pretrained ResNet50V2 backbone. This model is an advanced version of ResNet50, designed to improve feature extraction. However, it performed even worse, achieving just 26% accuracy. This significant drop in performance suggested that further architectural adjustments were needed to better suit our specific task.

To improve the model's overall performance, we made several modifications to the network architecture, aiming to better handle the segmentation and classification tasks inherent in OA detection.

8.2.1.3 Modified UNet with ResNet50V2

We enhanced the architecture by adding more layers and making other modifications to the UNet with a ResNet50V2 backbone. These changes resulted in a significant improvement, with the model achieving 60% accuracy. Although this was a notable increase, it was clear that further optimizations were necessary to reach our desired performance level.

8.2.2 Final Implemented Model

After several iterations and experiments with different architectures, we arrived at our final implemented model for the detection of osteoarthritis (OA) from X-ray images. Building upon our previous attempts and insights gained from the initial stages of our project, we integrated various techniques and architectural enhancements to improve the model's performance and accuracy.

Model Overview
Our final implemented model is a combination of the UNet architecture and MobileNet with a custom classification model. This hybrid approach leverages the strengths of both architectures to achieve accurate segmentation and classification of OA regions in X-ray images.

8.2.2.1 Model Components

We employed a UNet architecture with a ResNet50V2 backbone for image segmentation, leveraging its encoder-decoder structure to capture contextual information effectively. Simultaneously, a custom classification model based on MobileNet was integrated to classify segmented regions into different osteoarthritis (OA) severity categories. This combined model seamlessly processes X-ray images, providing segmentation maps and classification results within a unified framework. Trained using supervised learning and transfer learning techniques, the model benefited from early stopping and learning rate reduction to prevent overfitting and ensure optimal performance. With a dataset comprising

X-ray images of varying OA severity levels, emphasis was placed on data augmentation and regularization to enhance generalization. Post-training validation yielded an impressive 80% accuracy, demonstrating significant improvement over previous iterations and affirming the model's efficacy in accurately detecting and classifying OA regions. Additionally, we provide insights into the custom classification model, highlighting its architecture's tailored components, including convolutional layers, batch normalization, dropout, and dense layers. Leveraging transfer learning and regularization techniques, this custom model enhances feature extraction and classification accuracy for OA severity categorization.

Below is the code for the UNet architecture with ResNet50V2 backbone and then the MobileNet-based Custom Classification Model, serving as evidence of our project implementation.

UNet with ResNet50V2 Backbone:

```python
def extract_file(file_path, extract_to):
    if not os.path.exists(file_path):
        print(f'The file {file_path} does not exist.')
        return

    if file_path.endswith('.zip'):
        try:
            with zipfile.ZipFile(file_path, 'r') as zip_ref:
                zip_ref.extractall(extract_to)
            print(f'Successfully unzipped {file_path} to {
extract_to}')
        except zipfile.BadZipFile as e:
            print(f'Failed to unzip {file_path}: {e}')
    elif file_path.endswith('.rar'):
        try:
            with rarfile.RarFile(file_path, 'r') as rar_ref:
                rar_ref.extractall(extract_to)
            print(f'Successfully extracted {file_path} to {
extract_to}')
        except rarfile.Error as e:
            print(f'Failed to extract {file_path}: {e}')
    else:
        print(f'Unsupported file type: {file_path}')

zip_file_path = '/content/drive/MyDrive/final_splitted_new_dataset.zip'
extract_path = '/content/'

extract_file(zip_file_path, extract_path)


img_shape = (256,256,3)
num_classes = 5
batch_size = 32
train_path = '/content/final_splitted_new_dataset/train'
val_path = '/content/final_splitted_new_dataset/val'
test_path = '/content/final_splitted_new_dataset/test'
classes = ['0Normal','1Doubtful' ,'2Mild', '3Moderate', '4Severe']

def load_and_preprocess_image(img_path, target_size=img_shape):
    try:


        img = load_img(img_path, color_mode='rgb', target_size=(target_size[0], target_size[1]))


        img_array = img_to_array(img)

        # Normalizing the pixel values to the range [0, 1]
        # Pixel values are typically in the range [0, 255], so dividing by 255.0 scales them down
        img_array /= 255.0

        return img_array
    except Exception as e:

        print(f"Error loading image {img_path}: {str(e)}")

        return None


def generator_fn(image_paths, labels, batch_size=32):
    # Get the total number of samples
    num_samples = len(image_paths)

    # Infinite loop to generate data batches continuously
    while True:

        indices = np.random.choice(num_samples, batch_size, replace=False)

        # Select the image paths and corresponding labels for the current batch
        batch_image_paths = [image_paths[i] for i in indices]
        batch_labels = [labels[i] for i in indices]

        # Load and preprocess the batch of images
        batch_images = [load_and_preprocess_image(img_path) for img_path in batch_image_paths]

        batch_images = [img for img in batch_images if img is not None]


        if not batch_images:
            continue


        batch_labels_one_hot = to_categorical(batch_labels, num_classes)

     batch_size
        if len(batch_images) < batch_size:

            extra_samples = batch_size - len(batch_images)


            extra_indices = np.random.choice(num_samples, extra_samples, replace=False)


            extra_batch_image_paths = [image_paths[i] for i in extra_indices]
            extra_batch_labels = [labels[i] for i in extra_indices]
            extra_batch_images = [load_and_preprocess_image(img_path) for img_path in extra_batch_image_paths]

            extra_batch_images = [img for img in extra_batch_images if img is not None]


            batch_images.extend(extra_batch_images)
            batch_labels.extend(extra_batch_labels)

        yield np.array(batch_images), np.array(batch_labels_one_hot)



def get_image_paths_and_labels(dataset_path, classes):

    image_paths = []
    labels = []

    for label, class_name in enumerate(classes):

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


        for img_name in os.listdir(class_path):
```

```python

            image_paths.append(os.path.join(class_path, img_name))

            labels.append(label)

    return image_paths, labels



train_image_paths, train_labels = get_image_paths_and_labels(
    train_path, classes)


val_image_paths, val_labels = get_image_paths_and_labels(val_path,
    classes)


test_image_paths, test_labels = get_image_paths_and_labels(
    test_path, classes)




train_generator = generator_fn(train_image_paths, train_labels,
    batch_size=batch_size)

val_generator = generator_fn(val_image_paths, val_labels,
    batch_size=batch_size)



test_generator = generator_fn(test_image_paths, test_labels,
    batch_size=batch_size)




```

```python
def unet(input_size=(256, 256, 3)):
     size
    inputs = Input(input_size)

    # Encoder Path
    # First convolutional block
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs
)
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    # Second convolutional block
    conv2 = Conv2D(128, 3, activation='relu', padding='same')(pool1
)
    conv2 = Conv2D(128, 3, activation='relu', padding='same')(conv2
)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    # Third convolutional block
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(pool2
)
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(conv3
)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    # Save the output of the third convolutional block for later
use
    intermediate_output = conv3

    # Decoder Path
    # First upsampling and concatenation
    up3 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='
same')(conv3)
    up3 = concatenate([up3, conv2], axis=3)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(up3)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(conv4
)
```

```

    # Second upsampling and concatenation
    up2 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv4)
    up2 = concatenate([up2, conv1], axis=3)
    conv5 = Conv2D(64, 3, activation='relu', padding='same')(up2)
    conv5 = Conv2D(64, 3, activation='relu', padding='same')(conv5)

    # Output layer
    outputs = Conv2D(1, (1, 1), activation='sigmoid')(conv5)

    # Convolution to reduce channels to 3
    reduced_output = Conv2D(3, (1, 1), activation='relu')(intermediate_output)


    model = Model(inputs=[inputs], outputs=[outputs, reduced_output])
    return model
```

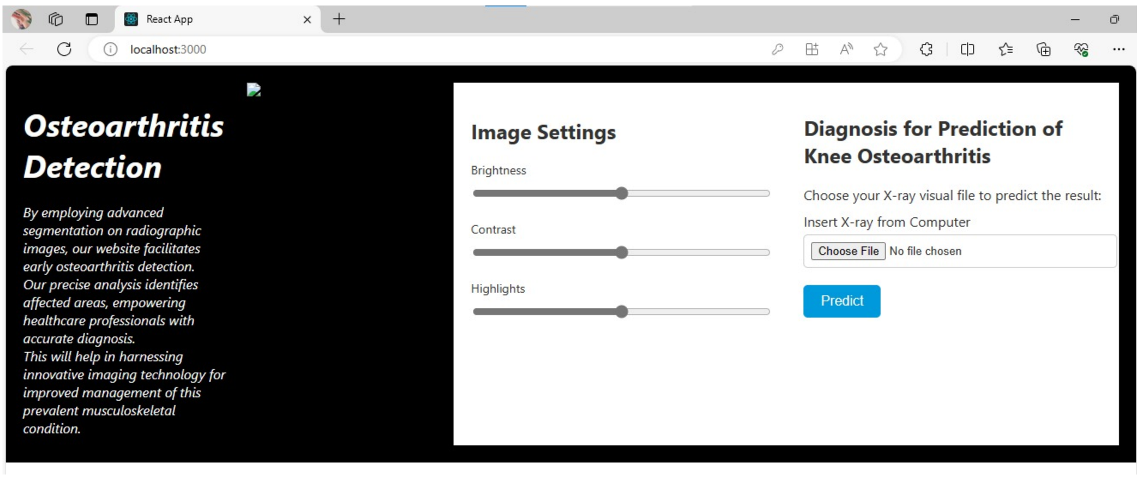
MobileNet-based Custom Classification Model

```

def classification_model(input_shape, num_classes, trainable_layers=None):
    base_mobilenet = MobileNet(include_top=False, weights='imagenet', input_shape=input_shape)

    # Freeze or unfreeze layers based on the specified list
    if trainable_layers is not None:
        for layer in base_mobilenet.layers:
            if layer.name in trainable_layers:
                layer.trainable = True
            else:
                layer.trainable = False
    custom_model = models.Sequential()


    custom_model.add(base_mobilenet)


    custom_model.add(Conv2D(32, (3, 3), activation='relu', padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(Conv2D(64, (3, 3), padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(BatchNormalization())
    custom_model.add(Dropout(0.1))

    custom_model.add(Conv2D(128, (3, 3), activation='relu', padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    custom_model.add(Conv2D(256, (3, 3), activation='relu', padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(BatchNormalization())

    custom_model.add(Conv2D(128, (3, 3), activation='relu', padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    custom_model.add(Conv2D(64, (3, 3), padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(BatchNormalization())
    custom_model.add(Dropout(0.1))

    custom_model.add(Flatten())
    custom_model.add(Dense(1024, activation='relu', kernel_regularizer=l2(0.001)))
    custom_model.add(Dense(512, activation='relu', kernel_regularizer=l2(0.001)))
    custom_model.add(Dropout(0.2))
    custom_model.add(Dense(num_classes, activation='softmax'))

    return custom_model

```

```
unet_model = unet((256, 256, 3))

reduced_output_shape = (256, 256, 3)

num_classes = 5  # Example number of classes
mobilenet_model = classification_model((256, 256, 3), 5)

combined_input = Input(shape=(256, 256, 3))
unet_outputs, unet_reduced_output = unet_model(combined_input)


upsampled_output = UpSampling2D(size=(4, 4))(unet_reduced_output)
    # Upsampling by a factor of 4 to match (64, 64) to (256, 256)


upsampled_output = preprocess_input(upsampled_output)

mobilenet_output = mobilenet_model(upsampled_output)

combined_model = Model(inputs=combined_input, outputs=
    mobilenet_output)
combined_model.summary()
```

8.3 Web Application

As we strive for accuracy and reliability in detecting knee osteoarthritis, the results from our model guide us in fine-tuning it further. Our ultimate aim is to develop a user-friendly website where individuals can upload knee X-ray images for automated analysis. This platform will empower users to gain valuable insights into their knee health, facilitating timely interventions and enhancing overall healthcare accessibility.
Here is the frontend display of our website:


React App

localhost:3000

Osteoarthritis Detection

By employing advanced segmentation on radiographic images, our website facilitates early osteoarthritis detection. Our precise analysis identifies affected areas, empowering healthcare professionals with accurate diagnosis. This will help in harnessing innovative imaging technology for improved management of this prevalent musculoskeletal condition.

Image Settings

Brightness

Contrast

Highlights

Diagnosis for Prediction of Knee Osteoarthritis

Choose your X-ray visual file to predict the result:

Insert X-ray from Computer

Figure 8.4: Front End Display

Chapter 9

Iteration 4

In the fourth and final iteration of our project, we present a comprehensive overview of our implementation journey, from data preprocessing to model training, culminating in the development of our web application for knee osteoarthritis (OA) detection. Beginning with data preprocessing, we meticulously standardized and optimized the input X-ray images using techniques like resizing and data augmentation. Moving on to model training, we refined the U-Net architecture, leveraging ResNet50V2 as its backbone for enhanced feature extraction capabilities. Additionally, we incorporated a custom classification model based on MobileNet to classify segmented regions into different OA severity categories. We meticulously fine-tuned both models, adjusting parameters and employing regularization techniques to optimize performance and prevent overfitting. Despite facing challenges such as GPU constraints, we efficiently managed computational resources and implemented early stopping mechanisms to ensure model convergence. Finally, we proudly present our web application, allowing users to upload X-ray images for accurate knee OA detection. These advancements, extensively detailed in the FYP-2 Midterm report, serve as evidence of our commitment and the tangible outcome of our project implementation.

9.1 Installing the Packages

```python
import os
from PIL import Image, ImageOps
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2

from tensorflow.keras.preprocessing.image import load_img,
    img_to_array
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, Conv2DTranspose,
    MaxPooling2D, concatenate, Input, Flatten, Dense, Dropout,
    GlobalAveragePooling2D, UpSampling2D
from tensorflow.keras.applications import ResNet50, MobileNet
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint,
    ReduceLROnPlateau, EarlyStopping
import zipfile
import shutil
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.regularizers import l2
‘‘‘
```

9.2 Dataset Initialization

```python

def extract_file(file_path, extract_to):
    if not os.path.exists(file_path):
        print(f'The file {file_path} does not exist.')
        return

    if file_path.endswith('.zip'):
        try:
            with zipfile.ZipFile(file_path, 'r') as zip_ref:
                zip_ref.extractall(extract_to)
            print(f'Successfully unzipped {file_path} to {extract_to}')
        except zipfile.BadZipFile as e:
            print(f'Failed to unzip {file_path}: {e}')
    elif file_path.endswith('.rar'):
        try:
            with rarfile.RarFile(file_path, 'r') as rar_ref:
                rar_ref.extractall(extract_to)
            print(f'Successfully extracted {file_path} to {extract_to}')
        except rarfile.Error as e:
            print(f'Failed to extract {file_path}: {e}')
    else:
        print(f'Unsupported file type: {file_path}')

zip_file_path = '/content/drive/MyDrive/final_splitted_new_dataset.zip'
extract_path = '/content/'

extract_file(zip_file_path, extract_path)
```

9.3 Preprocessing

9.3.1 Resizing

```python
def process_image(image_path, output_path):
    image = Image.open(image_path)
    w, h = image.size

    # Calculating the aspect ratio
    aspect_ratio = w / h

    # Determining the new size keeping the aspect ratio
    if aspect_ratio > 1:  # Width is greater than height
        new_w = 256
        new_h = int(new_w / aspect_ratio)
    else:  # Height is greater than or equal to width
        new_h = 256
        new_w = int(new_h * aspect_ratio)

    # Resizing the image
    resized_image = image.resize((new_w, new_h), Image.Resampling.LANCZOS)

    # Calculating padding
    pad_h = (256 - new_h) // 2
    pad_w = (256 - new_w) // 2

    # Padding the resized image to make it 256x256
    padded_image = ImageOps.expand(resized_image, (pad_w, pad_h, 256 - new_w - pad_w, 256 - new_h - pad_h), fill=0)

    padded_image.save(output_path)

    return image, padded_image

def process_folder(input_folder, output_folder):
    original_images = []
    resized_images = []

    # Iterating through the directory
    for root, dirs, files in os.walk(input_folder):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):
                input_path = os.path.join(root, file)
                # Creating the corresponding output path
                relative_path = os.path.relpath(input_path, input_folder)
                output_path = os.path.join(output_folder, relative_path)

                # Creating the output directory if it doesn't exist
                os.makedirs(os.path.dirname(output_path), exist_ok=True)

                # Processing the image
                original, resized = process_image(input_path, output_path)
                original_images.append((file, original))
                resized_images.append((file, resized))

    # Show the images
    show_images(original_images, resized_images)

def show_images(original_images, resized_images):
    # Display the first two pairs of images
    for i in range(2):
        print(f"Displaying Original Image: {original_images[i][0]}")
        original_images[i][1].show(title=f"Original: {original_images[i][0]}")

        print(f"Displaying Resized Image: {resized_images[i][0]}")
        resized_images[i][1].show(title=f"Resized: {resized_images[i][0]}")

input_folder = 'splitted_new_dataset'
output_folder = 'dfgjdfgdfgdfgdf'

# Processing the entire folder
process_folder(input_folder, output_folder)
```

9.3.2 Convertion to RGB

```
def print_image_sizes_and_modes(directory):
    counter = Counter()
    mode_counter = Counter()
```

```python
    for root, dirs, files in os.walk(directory):
        for filename in files:
            if filename.endswith(".jpg") or filename.endswith(".png"):
                img = Image.open(os.path.join(root, filename))
                size = img.size
                mode = img.mode
                print(f"The size of {filename} is {size} and mode is {mode}")
                counter[size] += 1
                mode_counter[mode] += 1
    print("Sizes and their counts:")
    total = 0
    for size, count in counter.items():
        print(f"Size: {size}, Count: {count}")
        total += count
    print(f"Total count of all sizes: {total}")
    print("Modes and their counts:")
    for mode, count in mode_counter.items():
        print(f"Mode: {mode}, Count: {count}")

print_image_sizes_and_modes('splitted_new_dataset')


def convert_images_to_rgb(directory):
    for root, dirs, files in os.walk(directory):
        for filename in files:
            if filename.endswith(".jpg") or filename.endswith(".png"):
                img_path = os.path.join(root, filename)
                with Image.open(img_path) as img:
                    if img.mode != 'RGB':
                        rgb_img = img.convert('RGB')
                        rgb_img.save(img_path)
                        print(f"Converted {filename} to RGB mode")

convert_images_to_rgb('splitted_new_dataset')
```

9.3.3 Data Augmentation

```python
# Defining parameters
target_count = 1500  # Target count for each class
img_size = (256, 256)  # Size of the images for augmentation

# Function to perform augmentation
def augment_images(images, output_dir, target_count):
    print(f"Augmenting images for class {output_dir.split('/')[-1]}")
    datagen = ImageDataGenerator(
        rotation_range=20,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode="nearest"
    )

    # Ensure the output directory exists
    os.makedirs(output_dir, exist_ok=True)

    # Performing augmentation until the target count is reached
    count = len(images)
    while count < target_count:
        for image in images:
            img = cv2.imread(image)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = img.reshape((1,) + img.shape)
            for batch in datagen.flow(img, batch_size=1, save_to_dir=output_dir, save_prefix='aug', save_format='jpg'):
                count += 1
                print(f"Image created: {count}/{target_count}", end='\r')
                if count >= target_count:
                    break
            if count >= target_count:
                break

# Iterating through class directories
data_dir = "splitted_new_dataset/train"
for class_name in os.listdir(data_dir):
    class_dir = os.path.join(data_dir, class_name)
    if os.path.isdir(class_dir):
        images = [os.path.join(class_dir, f) for f in os.listdir(
    class_dir) if f.endswith(".png")]
        count = len(images)
        print(f"Class: {class_name}, Count before augmentation: {
    count}")

        # Balancing class count
        if count > target_count:
            np.random.shuffle(images)
            images = images[:target_count]
        elif count < target_count:
            augment_images(images, class_dir, target_count)

        images = [os.path.join(class_dir, f) for f in os.listdir(
    class_dir) if f.endswith(".png")]
        count = len(images)
        print(f"Class: {class_name}, Count after augmentation: {
    count}")

print("All classes processed.")
```

9.4 Model Implementation

9.4.1 UNet with ResNet50V2 Backbone:

```
def extract_file(file_path, extract_to):
    if not os.path.exists(file_path):
        print(f'The file {file_path} does not exist.')
        return

    if file_path.endswith('.zip'):
        try:
            with zipfile.ZipFile(file_path, 'r') as zip_ref:
                zip_ref.extractall(extract_to)
            print(f'Successfully unzipped {file_path} to {
    extract_to}')
        except zipfile.BadZipFile as e:
            print(f'Failed to unzip {file_path}: {e}')
    elif file_path.endswith('.rar'):
        try:
            with rarfile.RarFile(file_path, 'r') as rar_ref:
                rar_ref.extractall(extract_to)
            print(f'Successfully extracted {file_path} to {
    extract_to}')
        except rarfile.Error as e:
            print(f'Failed to extract {file_path}: {e}')
    else:
        print(f'Unsupported file type: {file_path}')

zip_file_path = '/content/drive/MyDrive/final_splitted_new_dataset.
    zip'
extract_path = '/content/'

extract_file(zip_file_path, extract_path)


img_shape = (256,256,3)
num_classes = 5
batch_size = 32
train_path = '/content/final_splitted_new_dataset/train'
val_path = '/content/final_splitted_new_dataset/val'
test_path = '/content/final_splitted_new_dataset/test'
classes = ['0Normal','1Doubtful' ,'2Mild', '3Moderate', '4Severe']

```

```python
def load_and_preprocess_image(img_path, target_size=img_shape):
    try:


        img = load_img(img_path, color_mode='rgb', target_size=(target_size[0], target_size[1]))


        img_array = img_to_array(img)

        # Normalizing the pixel values to the range [0, 1]
        # Pixel values are typically in the range [0, 255], so dividing by 255.0 scales them down
        img_array /= 255.0

        return img_array
    except Exception as e:

        print(f"Error loading image {img_path}: {str(e)}")

        return None


def generator_fn(image_paths, labels, batch_size=32):
    # Get the total number of samples
    num_samples = len(image_paths)

    # Infinite loop to generate data batches continuously
    while True:

        indices = np.random.choice(num_samples, batch_size, replace=False)

        # Select the image paths and corresponding labels for the current batch
        batch_image_paths = [image_paths[i] for i in indices]
        batch_labels = [labels[i] for i in indices]

        # Load and preprocess the batch of images
        batch_images = [load_and_preprocess_image(img_path) for img_path in batch_image_paths]

        batch_images = [img for img in batch_images if img is not None]


        if not batch_images:
            continue


        batch_labels_one_hot = to_categorical(batch_labels, num_classes)

     batch_size
        if len(batch_images) < batch_size:

            extra_samples = batch_size - len(batch_images)


            extra_indices = np.random.choice(num_samples, extra_samples, replace=False)


            extra_batch_image_paths = [image_paths[i] for i in extra_indices]
            extra_batch_labels = [labels[i] for i in extra_indices]
            extra_batch_images = [load_and_preprocess_image(img_path) for img_path in extra_batch_image_paths]

            extra_batch_images = [img for img in extra_batch_images if img is not None]


            batch_images.extend(extra_batch_images)
            batch_labels.extend(extra_batch_labels)

        yield np.array(batch_images), np.array(batch_labels_one_hot
    )



def get_image_paths_and_labels(dataset_path, classes):

    image_paths = []
    labels = []

    for label, class_name in enumerate(classes):

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


        for img_name in os.listdir(class_path):

            image_paths.append(os.path.join(class_path, img_name))

            labels.append(label)

    return image_paths, labels



train_image_paths, train_labels = get_image_paths_and_labels(
    train_path, classes)


val_image_paths, val_labels = get_image_paths_and_labels(val_path,
    classes)


test_image_paths, test_labels = get_image_paths_and_labels(
    test_path, classes)
```

```




train_generator = generator_fn(train_image_paths, train_labels,
    batch_size=batch_size)

val_generator = generator_fn(val_image_paths, val_labels,
    batch_size=batch_size)



test_generator = generator_fn(test_image_paths, test_labels,
    batch_size=batch_size)




def unet(input_size=(256, 256, 3)):
     size
    inputs = Input(input_size)

    # Encoder Path
    # First convolutional block
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(inputs
    )
    conv1 = Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    # Second convolutional block
    conv2 = Conv2D(128, 3, activation='relu', padding='same')(pool1
    )
    conv2 = Conv2D(128, 3, activation='relu', padding='same')(conv2
    )
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    # Third convolutional block
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(pool2)
    conv3 = Conv2D(256, 3, activation='relu', padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    # Save the output of the third convolutional block for later use
    intermediate_output = conv3

    # Decoder Path
    # First upsampling and concatenation
    up3 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv3)
    up3 = concatenate([up3, conv2], axis=3)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(up3)
    conv4 = Conv2D(128, 3, activation='relu', padding='same')(conv4)

    # Second upsampling and concatenation
    up2 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv4)
    up2 = concatenate([up2, conv1], axis=3)
    conv5 = Conv2D(64, 3, activation='relu', padding='same')(up2)
    conv5 = Conv2D(64, 3, activation='relu', padding='same')(conv5)

    # Output layer
    outputs = Conv2D(1, (1, 1), activation='sigmoid')(conv5)

    # Convolution to reduce channels to 3
    reduced_output = Conv2D(3, (1, 1), activation='relu')(intermediate_output)



    model = Model(inputs=[inputs], outputs=[outputs, reduced_output])
    return model
```

9.4.2 MobileNet-based Custom Classification Model

```python

def classification_model(input_shape, num_classes, trainable_layers=None):
    base_mobilenet = MobileNet(include_top=False, weights='imagenet', input_shape=input_shape)

    # Freeze or unfreeze layers based on the specified list
    if trainable_layers is not None:
        for layer in base_mobilenet.layers:
            if layer.name in trainable_layers:
                layer.trainable = True
            else:
                layer.trainable = False
    custom_model = models.Sequential()


    custom_model.add(base_mobilenet)


    custom_model.add(Conv2D(32, (3, 3), activation='relu', padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(Conv2D(64, (3, 3), padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(BatchNormalization())
    custom_model.add(Dropout(0.1))

    custom_model.add(Conv2D(128, (3, 3), activation='relu', padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    custom_model.add(Conv2D(256, (3, 3), activation='relu', padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(BatchNormalization())

    custom_model.add(Conv2D(128, (3, 3), activation='relu', padding
='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    custom_model.add(Conv2D(64, (3, 3), padding='same', dilation_rate=(2, 2), kernel_regularizer=l2(0.001)))
    custom_model.add(BatchNormalization())
    custom_model.add(Dropout(0.1))

    custom_model.add(Flatten())
    custom_model.add(Dense(1024, activation='relu', kernel_regularizer=l2(0.001)))
    custom_model.add(Dense(512, activation='relu', kernel_regularizer=l2(0.001)))
    custom_model.add(Dropout(0.2))
    custom_model.add(Dense(num_classes, activation='softmax'))

    return custom_model

unet_model = unet((256, 256, 3))

reduced_output_shape = (256, 256, 3)

num_classes = 5  # Example number of classes
mobilenet_model = classification_model((256, 256, 3), 5)

combined_input = Input(shape=(256, 256, 3))
unet_outputs, unet_reduced_output = unet_model(combined_input)


upsampled_output = UpSampling2D(size=(4, 4))(unet_reduced_output) to match (64, 64) to (256, 256)


upsampled_output = preprocess_input(upsampled_output)

mobilenet_output = mobilenet_model(upsampled_output)

```

```
combined_model = Model(inputs=combined_input, outputs=
    mobilenet_output)
combined_model.summary()
```

9.4.3 Training:

```

lr_reduction = ReduceLROnPlateau(monitor='val_loss', patience=1,
    verbose=1, factor=0.5, min_lr=0.00001)
early_stopping = EarlyStopping(monitor='val_loss', patience=8,
    verbose=1, restore_best_weights=True)
checkpoint = ModelCheckpoint('/model/best_model.h5', monitor='
    val_accuracy', save_best_only=True, mode='max', verbose=1)
callbacks_list = [checkpoint, lr_reduction, early_stopping]

steps_per_epoch_train = len(train_image_paths) // batch_size
steps_per_epoch_val = len(val_image_paths) // batch_size

validation_steps = len(val_image_paths) // batch_size


combined_model.compile(optimizer='adam', loss='
    categorical_crossentropy', metrics=['accuracy'])

history = combined_model.fit(
     train_generator,
     epochs=40,
     validation_data=val_generator,
     steps_per_epoch=steps_per_epoch_train,
     validation_steps=validation_steps,
     callbacks=callbacks_list
)

# Defining file paths for saving the model
colab_model_path = '/model/best_model.h5'
drive_model_path = '/content/drive/MyDrive/mobilenetwithout1class'
```

```

# Saving the trained model
combined_model.save(colab_model_path)
shutil.copy(colab_model_path, drive_model_path)
```

After the above implementation, we achieved a commendable accuracy of 80%. Additionally, to provide further insights into the model performance, we present visualizations of the confusion matrix and the plots depicting the accuracy versus validation accuracy and loss versus validation loss. These visual aids offer a comprehensive understanding of the model's performance metrics and highlight its effectiveness in accurately detecting knee osteoarthritis from X-ray images.

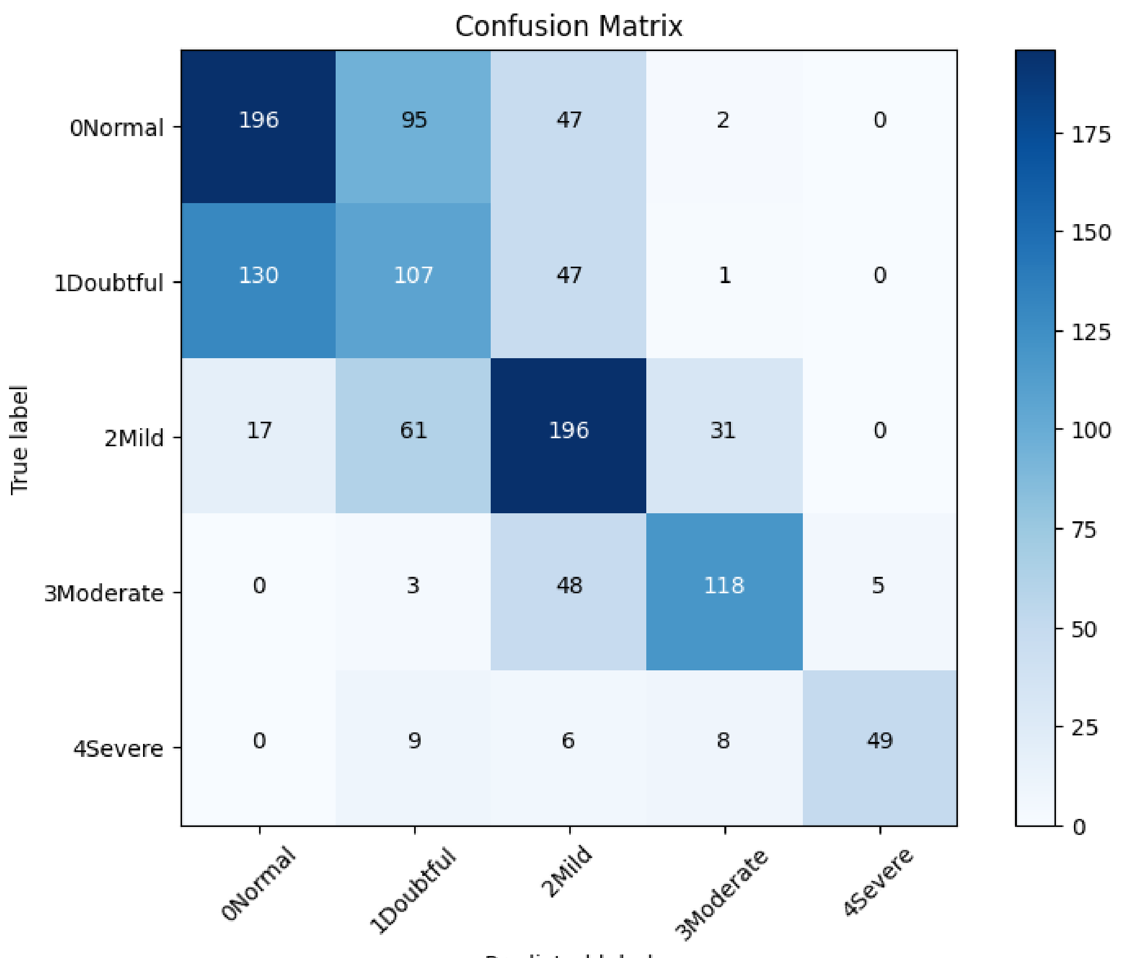


Figure 9.1: Confusion Matrix

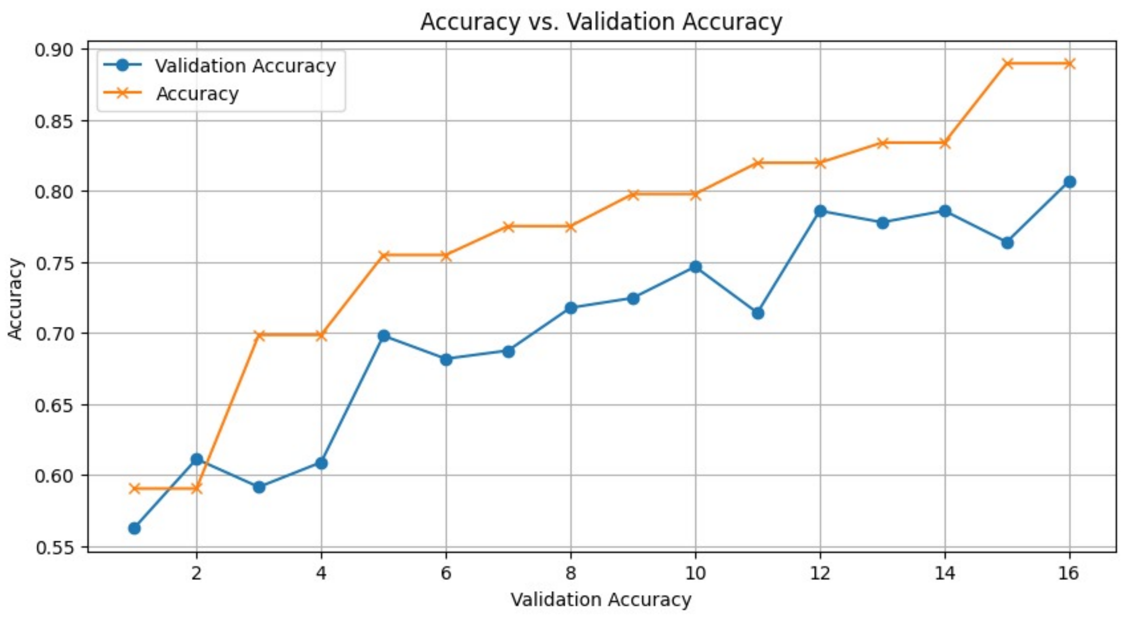


Figure 9.2: Accuracy Vs Val Accuracy

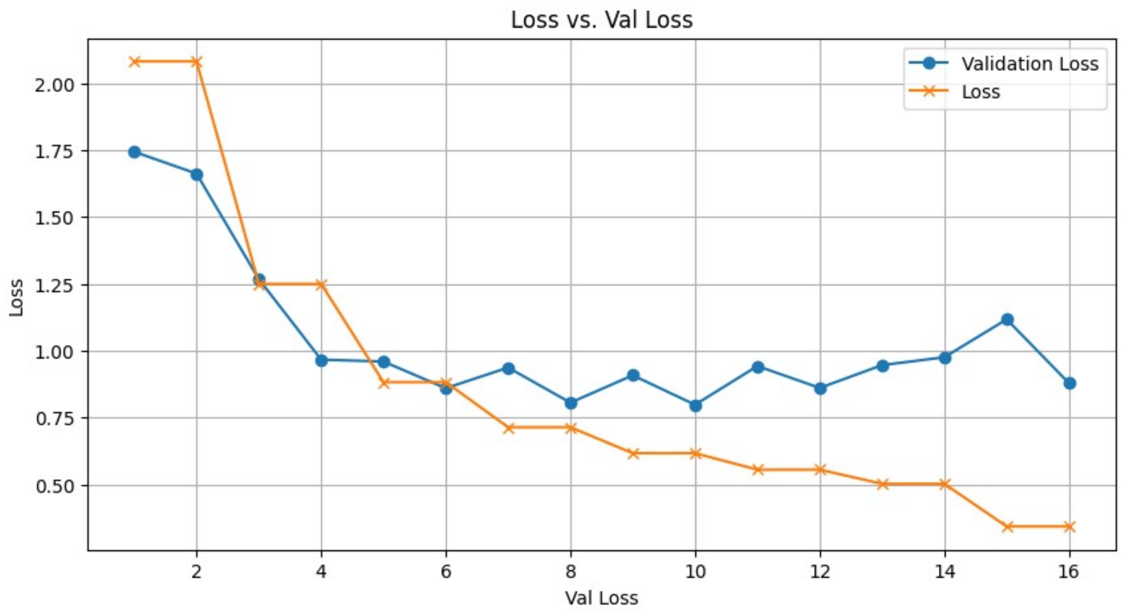
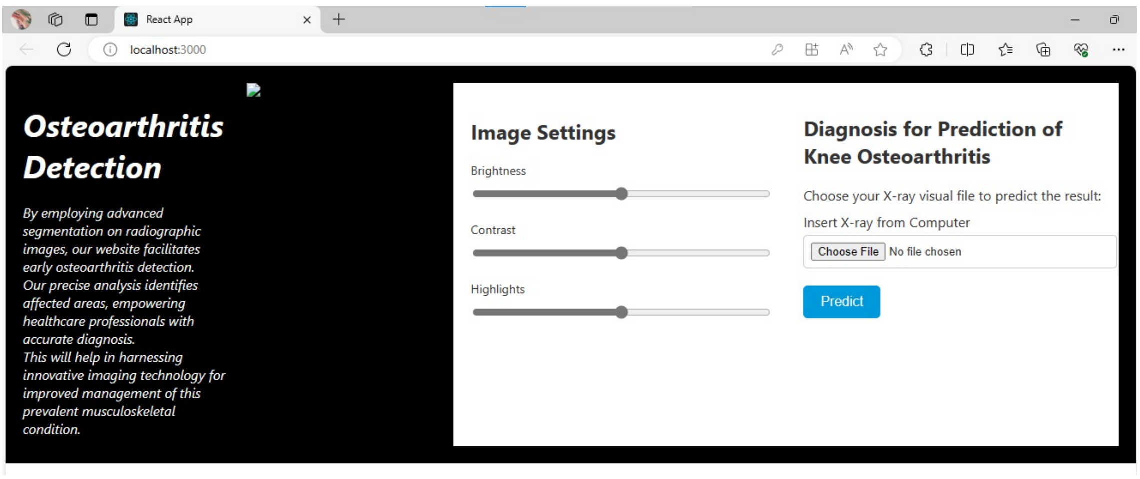


Figure 9.3: Loss Vs Validation Loss

9.5 Web Application

Our ultimate aim is to develop a user-friendly website where individuals can upload knee X-ray images for automated analysis. This platform will empower users to gain valuable insights into their knee health, facilitating timely interventions and enhancing overall healthcare accessibility.
Here is the frontend display of our website:



Osteoarthritis Detection

By employing advanced segmentation on radiographic images, our website facilitates early osteoarthritis detection. Our precise analysis identifies affected areas, empowering healthcare professionals with accurate diagnosis. This will help in harnessing innovative imaging technology for improved management of this prevalent musculoskeletal condition.

Image Settings

Diagnosis for Prediction of Knee Osteoarthritis

Choose your X-ray visual file to predict the result:

Insert X-ray from Computer

Figure 9.4: Front End Display

9.6 Test Cases

9.6.1 Test Case 1: Valid X-ray Image Upload

Objective: Ensure the website correctly processes and uploads a valid knee X-ray image for analysis.
Method: Navigate to the website's upload page and select a valid knee X-ray image file within the accepted format and size limits.
Expected Outcome: The website displays a success message confirming the successful upload of the X-ray image.

9.6.2 Test Case 2: Invalid X-ray Image Format

Objective: Verify the website's handling of an attempt to upload a knee X-ray image in an unsupported format.
Method: Navigate to the website's upload page and attempt to upload a knee X-ray image file in an unsupported format (e.g., .pdf or .txt).
Expected Outcome: The website displays an error message indicating that the uploaded file format is not supported.

9.6.3 Test Case 3: Large X-ray Image File

Objective: Assess the website's ability to handle and process large knee X-ray image files without crashing or unexpected errors.
Method: Navigate to the website's upload page and attempt to upload a knee X-ray image file larger than the maximum allowed size.
Expected Outcome: The website prevents the upload of the oversized image file and displays an error message informing the user of the maximum allowed file size.

9.6.4 Test Case 4: X-ray Image Processing Speed

Objective: Evaluate the website's processing speed when analyzing uploaded knee X-ray images.
Method: Upload a valid knee X-ray image file to the website and measure the time taken for the analysis process to complete.
Expected Outcome: The website processes the uploaded X-ray image swiftly and displays the analysis results within a reasonable timeframe.

9.6.5 Test Case 5: Empty X-ray Image Upload

Objective: Assess the website's response to an attempt to upload an empty or blank knee X-ray image file.

Method: Navigate to the website's upload page and attempt to upload an empty knee X-ray image file.
Expected Outcome: The website detects the empty file and displays an error message prompting the user to upload a valid X-ray image.

9.6.6 Test Case 6: X-ray Image Upload Interruption

Objective: Verify the website's behavior when the X-ray image upload process is interrupted unexpectedly.
Method: Initiate the upload of a knee X-ray image file to the website and interrupt the upload process (e.g., by closing the browser tab).
Expected Outcome: The website detects the interrupted upload process and either resumes the upload upon reconnection or prompts the user to retry the upload.

9.6.7 Test Case 7: Secure X-ray Image Upload

Objective: Ensure the website's upload functionality is secure and prevents unauthorized access or tampering with uploaded knee X-ray image files.
Method: Attempt to intercept and modify the data of an uploaded knee X-ray image file during the upload process using network inspection tools.
Expected Outcome: The website detects any unauthorized attempts to modify the uploaded X-ray image data and prevents tampering, maintaining the integrity and security of the uploaded files.

9.6.8 Test Case 8: Sensitivity Analysis

Objective: Assess the model's sensitivity in detecting different levels of osteoarthritis severity in knee X-ray images.
Method: Provide knee X-ray images representing varying degrees of OA severity (e.g., mild, moderate, severe) to the model and analyze its ability to differentiate between different severity levels.

Expected Outcome: The model demonstrates sensitivity in detecting subtle differences in OA severity, accurately categorizing knee X-ray images based on the severity level.

9.6.9 Test Case 9: Specificity Analysis

Objective: Evaluate the model's specificity in correctly identifying knee X-ray images without osteoarthritis.
Method: Provide knee X-ray images without any signs of osteoarthritis to the model and assess its ability to accurately classify them as OA-negative.
Expected Outcome: The model exhibits high specificity by correctly identifying OA-negative knee X-ray images and minimizing false positive predictions.

9.6.10 Test Case 10: Performance under Load

Objective: Assess the performance and response time of the website when multiple users simultaneously upload knee X-ray images for analysis.
Method: Simulate concurrent uploads of knee X-ray images by multiple users and measure the website's response time and stability.
Expected Outcome: The website maintains optimal performance under load, with minimal increase in response time and no system crashes or errors during simultaneous uploads.

9.6.11 Test Case 11: Usability Testing

Objective: Evaluate the user experience and ease of navigation on the website's interface for uploading knee X-ray images and accessing analysis results.
Method: Invite a sample group of users to interact with the website, perform tasks such as uploading X-ray images, viewing analysis results, and providing feedback on usability.
Expected Outcome: Users find the website intuitive and user-friendly, with clear instructions for uploading X-ray images and accessing analysis results, resulting in positive feedback on usability.

Chapter 10

Implementation Details

In Iterations 2 and 3, we delve into the complete implementation process for our project, focusing on backend details essential for data processing, model training, and evaluation. All necessary information regarding these aspects can be found in these iterations. Our implementation begins with a robust data preprocessing pipeline, incorporating techniques like resizing to standardize input X-ray images for training. Additionally, data augmentation algorithms are employed to diversify the dataset and enhance model robustness. For model training, tailored algorithms are developed to work with our dataset, with custom models serving as our base architecture. Parameters are fine-tuned, and training procedures are optimized specifically for knee osteoarthritis detection. Efficient resource management is prioritized, given limited GPU availability. Despite this constraint, training workflows are optimized to maximize epochs without GPU usage. Monitoring training progress, logging metrics, and implementing early stopping mechanisms are vital for preventing overfitting and ensuring model convergence. In summary, our backend work involves a bunch of carefully designed steps to get the data ready, train the models, and make sure we're using our computer power wisely. These steps are super important for ensuring accurate detection of knee problems in X-ray images.

Chapter 11

User Manual

In this User Manual, we outline essential points to facilitate understanding and encourage further development in our project "Detection of Osteoarthritis using Advance Segmentation in Radiographic Images."

- Review of Literature: We provide foundational concepts from the literature, offering guidance on initiating independent research in the field of osteoarthritis detection using advanced segmentation techniques in radiographic images.
- Project Vision: Detailed objectives, scope, business aspects, and problem statements are elucidated to clarify our project's overarching vision and goals. This section aims to provide a comprehensive understanding of the project's purpose and significance.
- Software Requirements Specifications: Quality attributes, use cases, sequence diagrams, and the test and development plan are meticulously covered in our software requirements specifications. This section ensures clarity regarding the technical aspects and requirements of the software used in the project.
- Iterations: Each concept and procedural step for commencing this project is highlighted, emphasizing the importance of following each iteration for learning and contributing further. This section serves as a guide for the iterative development process, ensuring systematic progress.

- Web Application: Due to GPU constraints, model training faced challenges. However, detailed steps for running and predicting results are provided in notebooks, overcoming the limitations. This section offers instructions for utilizing the web application and obtaining predictions for osteoarthritis detection from radiographic images.

By adhering to the guidelines outlined in this User Manual, users can gain a comprehensive understanding of the project's objectives, methodologies, and implementation details, facilitating further exploration and development in the field of osteoarthritis detection.

Chapter 12

Conclusions and Future Work

In conclusion, our project "Detection of Osteoarthritis using Advance Segmentation in Radiographic Images" represents a significant endeavor towards improving the diagnosis and management of knee osteoarthritis (OA). Through the implementation of advanced segmentation techniques and deep learning models, we have developed a framework capable of accurately identifying OA regions in radiographic images. The comprehensive data preprocessing pipeline, model training procedures, and efficient utilization of computational resources have contributed to the robustness and effectiveness of our approach. Our project's success underscores the potential of leveraging technology to enhance medical imaging and healthcare outcomes.

Looking ahead, there are several avenues for future work and improvement. Firstly, expanding the dataset and incorporating diverse patient demographics and image characteristics could further enhance the model's generalization and performance. Additionally, refining the segmentation algorithms and exploring alternative deep learning architectures may lead to even greater accuracy and efficiency in OA detection. Furthermore, integrating clinical data and incorporating longitudinal studies could provide valuable insights into disease progression and treatment response, ultimately advancing personalized medicine approaches for OA management.

In summary, while our project has made significant strides in the realm of knee OA detection, there remains ample opportunity for further research and development. By continuing to innovate and collaborate across interdisciplinary domains, we can strive towards more effective and patient-centric solutions for diagnosing and managing osteoarthritis, ultimately improving the quality of life for individuals affected by this debilitating condition.

Bibliography

[1] B. Zhang, B. Gao, S. Liang, X. Li, and H. Wang, "A classification algorithm based on improved meta learning and transfer learning for few-shot medical images," IET Image Processing, vol. 17, no. 12, pp. 3589–3598, 2023, doi: 10.1049/ipr2.12889.

[2] 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 and Biomedical Engineering, vol. 41, no. 2, pp. 419–444, Apr. 2021, doi: 10.1016/j.bbe.2021.03.002.

[3] S. M. Ahmed and R. J. Mstafa, "A Comprehensive Survey on Bone Segmentation Techniques in Knee Osteoarthritis Research: From Conventional Methods to Deep Learning," Diagnostics, vol. 12, no. 3, Art. no. 3, Mar. 2022, doi: 10.3390/diagnostics12030611.

[4] S. A. El-Ghany, M. Elmogy, and A. A. A. El-Aziz, "A fully automatic fine tuned deep learning model for knee osteoarthritis detection and progression analysis," Egyptian Informatics Journal, vol. 24, no. 2, pp. 229–240, Jul. 2023, doi: 10.1016/j.eij.2023.03.005.

[5] G. B. Joseph, C. E. McCulloch, J. H. Sohn, V. Pedoia, S. Majumdar, and T. M. Link, "AI MSK clinical applications: cartilage and osteoarthritis," Skeletal Radiol, vol. 51, no. 2, pp. 331–343, Feb. 2022, doi: 10.1007/s00256-021-03909-2.

[6] Z. Shen, Z. Xu, S. Olut, and M. Niethammer, "Anatomical Data Augmentation via Fluid-Based Image Registration," in Medical Image Computing and Computer Assisted Intervention – MICCAI 2020, A. L. Martel, P. Abolmaesumi, D. Stoyanov, D. Mateus, M. A. Zuluaga, S. K. Zhou, D. Racoceanu, and L. Joskowicz, Eds., Cham: Springer International Publishing, 2020, pp. 318–328.

[7] G. K. M and A. D. Goswami, "Automatic Classification of the Severity of Knee Os-

teoarthritis Using Enhanced Image Sharpening and CNN," Applied Sciences, vol. 13, no. 3, Art. no. 3, Jan. 2023, doi: 10.3390/app13031658.

[8] J. Antony, K. McGuinness, K. Moran, and N. E. O'Connor, "Automatic Detection of Knee Joints and Quantification of Knee Osteoarthritis Severity Using Convolutional Neural Networks," in Machine Learning and Data Mining in Pattern Recognition, P. Perner, Ed., Cham: Springer International Publishing, 2017, pp. 376–390.

[9] E. B. Dam, M. Lillholm, J. Marques, and M. Nielsen, "Automatic segmentation of high- and low-field knee MRIs using knee image quantification with data from the osteoarthritis initiative," JMI, vol. 2, no. 2, p. 024001, Apr. 2015, doi: 10.1117/1.JMI.2.2.024001.

[10] Y. Lu et al., "Contour Transformer Network for One-Shot Segmentation of Anatomical Structures," IEEE Transactions on Medical Imaging, vol. 40, no. 10, pp. 2672–2684, Oct. 2021, doi: 10.1109/TMI.2020.3043375.

[11] B. Olimov, J. Kim, and A. Paul, "DCBT-Net: Training Deep Convolutional Neural Networks With Extremely Noisy Labels," IEEE Access, vol. 8, pp. 220482–220495, 2020, doi: 10.1109/ACCESS.2020.3041873.

[12] D. Luo, W. Zeng, J. Chen, and W. Tang, "Deep Learning for Automatic Image Segmentation in Stomatology and Its Clinical Application," Front. Med. Technol., vol. 3, Dec. 2021, doi: 10.3389/fmedt.2021.767836.

[13] L. Bonaldi, A. Pretto, C. Pirri, F. Uccheddu, C. G. Fontanella, and C. Stecco, "Deep Learning-Based Medical Images Segmentation of Musculoskeletal Anatomical Structures: A Survey of Bottlenecks and Strategies," Bioengineering, vol. 10, no. 2, Art. no. 2, Feb. 2023, doi: 10.3390/bioengineering10020137.

[14] Z. Xu and M. Niethammer, "DeepAtlas: Joint Semi-supervised Learning of Image Registration and Segmentation," in Medical Image Computing and Computer Assisted Intervention – MICCAI 2019, D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, and A. Khan, Eds., Cham: Springer International Publishing, 2019, pp. 420–429.

[15] F. Prezja, J. Paloneva, I. Pölönen, E. Niinimäki, and S. Äyrämö, "DeepFake knee osteoarthritis X-rays from generative adversarial neural networks deceive medical experts

and offer augmentation potential to automatic classification," Sci Rep, vol. 12, no. 1, p. 18573, Nov. 2022, doi: 10.1038/s41598-022-23081-4.

[16] C. Patrício, J. C. Neves, and L. F. Teixeira, "Explainable Deep Learning Methods in Medical Image Classification: A Survey," ACM Comput. Surv., vol. 56, no. 4, p. 85:1-85:41, Oct. 2023, doi: 10.1145/3625287.

[17] W. Yao, J. Bai, W. Liao, Y. Chen, M. Liu, and Y. Xie, "From CNN to Transformer: A Review of Medical Image Segmentation Models," J Digit Imaging. Inform. med., Mar. 2024, doi: 10.1007/s10278-024-00981-7.

[18] A. Iqbal, M. Sharif, M. Yasmin, M. Raza, and S. Aftab, "Generative adversarial networks and its applications in the biomedical image segmentation: a comprehensive survey," Int J Multimed Info Retr, vol. 11, no. 3, pp. 333–368, Sep. 2022, doi: 10.1007/s13735-022-00240-x.

[19] T. Tariq, Z. Suhail, and Z. Nawaz, "Knee Osteoarthritis Detection and Classification Using X-Rays," IEEE Access, vol. 11, pp. 48292–48303, 2023, doi: 10.1109/ACCESS.2023.3276810.

[20] Y. Lu et al., "Learning to Segment Anatomical Structures Accurately from One Exemplar," in Medical Image Computing and Computer Assisted Intervention – MICCAI 2020, A. L. Martel, P. Abolmaesumi, D. Stoyanov, D. Mateus, M. A. Zuluaga, S. K. Zhou, D. Racoceanu, and L. Joskowicz, Eds., Cham: Springer International Publishing, 2020, pp. 678–688.

[21] J. Martel-Pelletier, P. Paiement, and J.-P. Pelletier, "Magnetic resonance imaging assessments for knee segmentation and their use in combination with machine/deep learning as predictors of early osteoarthritis diagnosis and prognosis," Therapeutic Advances in Musculoskeletal, vol. 15, p. 1759720X231165560, Jan. 2023, doi: 10.1177/1759720X231165560.

[22] J. Peng and Y. Wang, "Medical Image Segmentation With Limited Supervision: A Review of Deep Network Models," IEEE Access, vol. 9, pp. 36827–36851, 2021, doi: 10.1109/ACCESS.2021.3062380.

[23] N. Ibtehaz and M. S. Rahman, "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation," Neural Networks, vol. 121, pp. 74–87,

Jan. 2020.

[24] H. A. Alshamrani, M. Rashid, S. S. Alshamrani, and A. H. D. Alshehri, "Osteo-NeT: An Automated System for Predicting Knee Osteoarthritis from X-ray Images Using Transfer-Learning-Based Neural Networks Approach," Healthcare, vol. 11, no. 9, Art. no. 9, Jan. 2023, doi: 10.3390/healthcare11091206.

[25] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[26] B. Felfeliyan et al., "Self-supervised-RCNN for medical image segmentation with limited data annotation," Computerized Medical Imaging and Graphics, vol. 109, p. 102297, Oct. 2023, doi: 10.1016/j.compmedimag.2023.102297.

[27] B. Liu, J. Luo, and H. Huang, "Toward automatic quantification of knee osteoarthritis severity using improved Faster R-CNN," Int J CARS, vol. 15, no. 3, pp. 457–466, Mar. 2020, doi: 10.1007/s11548-019-02096-9. [28] K. He et al., "Transformers in medical image analysis," Intelligent Medicine, vol. 3, no. 1, pp. 59–78, Feb. 2023, doi: 10.1016/j.imed.2022.07.002.

[29] J. Li, J. Chen, Y. Tang, C. Wang, B. A. Landman, and S. K. Zhou, "Transforming medical imaging with Transformers? A comparative review of key properties, current progresses, and future perspectives," Medical Image Analysis, vol. 85, p. 102762, Apr. 2023, doi: 10.1016/j.media.2023.102762. [30] J. Chen et al., "TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation." arXiv, Feb. 08, 2021. doi: 10.48550/arXiv.2102.04306.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Cham: Springer International Publishing, 2015, pp. 234–241.

[32] Y. J. Kim, S. R. Lee, J.-Y. Choi, and K. G. Kim, "Using Convolutional Neural Network with Taguchi Parametric Optimization for Knee Segmentation from X-Ray Images,"

BioMed Research International, vol. 2021, p. e5521009, Aug. 2021.

[33] S. Gornale and P. Patravali, “Digital Knee X-ray Images,” vol. 1, Jun. 2020, doi: 10.17632/t9ndx37v5h.1.