A Project Report On

KNEE OSTEOARTHRITIS DETECTION AND ITS SEVERITY USING DEEP LEARNING

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

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted By

V NAVYA VENKATA SAI	**19P31A1253**
ANKIT KUMAR	**19P31A1203**
K HARI CHAITANYA	**19P31A1219**
V BALAJI	**19P31A1254**

Under the esteemed supervision of

Mr. M M Siva Krishna, M.Tech

Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Permanently Affiliated to JNTUK, Kakinada * Approved by AICTE New Delhi

Accredited by NBA, Accredited by NAAC (A+) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh.

2019-2023

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Permanently Affiliated to JNTUK, Kakinada * Approved by AICTE New Delhi

Accredited by NBA, Accredited by NAAC (A+) with 3.4 CGPA

Aditya Nagar, ADB Road, Surampalem , Kakinada District, Andhra Pradesh

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project work entitled "**Knee Osteoarthritis Detection And Its Severity Using Deep Learning**", is a bonafide work carried out by **V Navya Venkata Sai (19P31A1253), Ankit Kumar (19P31A1203), K Hari Chaitanya (19P31A1219), V Balaji (19P31A1254)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology** from Aditya College of Engineering & Technology during the academic year 2019-2023.

Project Guide	**Head Of The Department**
Mr. M M Siva Krishna, M.Tech	**Mr. R V V N Bheema Rao.,M.Tech.,(Ph.D)**
Assistant Professor	**Assistant Professor**

EXTERNAL EXAMINER

DECLARATION

We here by declare that this project entitled "**Knee Osteoarthritis Detection And Its Severity Using Deep Learning**", has been undertaken by us and this work has been submitted to **Aditya College of Engineering & Technology** affiliated to JNTUK, Kakinada, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology**.

We further declare that this project work has not been submitted in full or part for the award of any degree of this or in any other educational institutions.

Project Associates

V Navya Venkata Sai	**19P31A1253**
Ankit Kumar	**19P31A1203**
K Hari Chaitanya	**19P31A1219**
V Balaji	**19P31A1254**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to our Project Supervisor, **Mr. M M Siva Krishna, M.Tech** who has guided us a lot and encouraged us in every step of the project work, his valuable moral support and guidance throughout the project helped us to a great extent.

We wish to express our sincere thanks to the Head of the Department **Mr. R V V N Bheema Rao M.Tech.,(Ph.D)** for his valuable guidance given to us throughout the period of the project work and throughout the program.

We feel elated to thank **Dr. A Rama Krishna Ph.D** Dean of Aditya College of Engineering & Technology for his cooperation in completion of our project and throughout the program.

We feel elated to thank **Dr. Dola Sanjay S Ph.D** Principal of Aditya College of Engineering & Technology for his cooperation in completion of our project and throughout the program.

We wish to express our sincere thanks to all **faculty members, lab programmers** for their valuable assistance throughout the period of the project.

We avail this opportunity to express our deep sense and heart full thanks to the Management of **Aditya College Of Engineering & Technology** for providing a great support for us in completing our project and also throughout the program.

V Navya Venkata Sai	**(19P31A1253)**
Ankit Kumar	**(19P31A1203)**
K Hari Chaitanya	**(19P31A1219)**
V Balaji	**(19P31A1254)**

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Institute Vision & Mission

Vision

To induce higher planes of learning by imparting technical education with

- International standards
- Applied research
- Creative Ability
- Values based instruction and to emerge as a premiere institute

Mission

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- Innovative research and development
- Industry institute interaction
- Empowered manpower

Principal

PRINCIPAL
Aditya College of
Engineering & Technolo
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Vision

To be a department with high repute and focused on quality education

Mission

- To Provide an environment for the development of professionals with knowledge and skills
- To promote innovative learning
- To promote innovative ideas towards society
- To foster trainings with institutional collaborations
- To involve in the development of software applications for societal needs

[illegible]

Head of the Department

Head of the Department
Dept.of IT
Aditya College of Engineering & Technology
SURAMPALEM 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Program Educational Objectives

Program educational objectives are broad statements that describe the career and professional accomplishments that the program is preparing graduates to achieve.

PEO-1:

Graduates will be skilled in Mathematics, Science & modern engineering tools to solve real life problems.

PEO-2:

Excel in the IT industry with the attained knowledge and skills or pursue higher studies to acquire emerging technologies and become an entrepreneur.

PEO-3:

Accomplish a successful career and nurture as a responsible professional with ethics and human values.

[illegible]

Head of the Department

Head of the Department
Dept.of IT
Aditya College of Engineering & Technology
SURAMPALEM - 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Program Outcomes

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

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

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

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

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

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

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

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

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

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

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

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

[illegible]

Head of the Department

Head of the Department
Dept.of IT
Aditya College of Engineering & Technology
SURAMPALEM 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technol
SURAMPALEM

Aditya College of Engineering & Technology

Approved by AICTE, New Delhi, * Permanently Affiliated to JNTUK, Kakinada
Accredited by NBA, Accredited by NAAC (A+) with CGPA of 3.4
Recognized by UGC under Section 2(f) and 12(B) of UGC Act 1956
Aditya Nagar, ADB Road, Surampalem

Department of Information Technology

Program Specific Outcomes

PSO-1:

Apply mathematical foundations, algorithmic and latest computing tools and techniques to design computer-based systems to solve engineering problems.

PSO-2:

Apply knowledge of engineering and develop software-based applications for research and development in the areas of relevance under realistic constraints.

PSO-3:

Apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product.

[illegible]

Head of the Department

Head of the Department
Dept.of IT
Aditya College of Engineering & Technology
SURAMPALEM 533 437

[illegible]

Principal

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Surampalem, Andhra Pradesh.

DEPARTMENT OF INFORMATION TECHNOLOGY

<u>Course Outcomes (COs)</u>

Course Name:	Project work	**Year**	IV
Faculty Name:	Mr. M M Siva Krishna, M.Tech	**Semester**	II
Academic year:	2022-23	**Regulation**	R19

Course Outcomes

After completing this course, the student will be able to:

CO Number	CO Statement	Taxonomy
CO1	**Apply** fundamental and disciplinary concepts and methods in ways appropriate to their principal areas of study.	Apply
CO2	**Demonstrate** skill and knowledge of current information and technological tools and techniques specific to the professional field of study.	Apply
CO3	**Effectively** communicate with engineers and the community at large in written, oral and visual forms.	Apply
CO4	**Apply** the theoretical knowledge to identify, analyze and solve industrial problems with team work and multidisciplinary approach.	Apply
CO5	**Demonstrate** professionalism with ethics and relate engineering issues to broader societal context.	Apply
CO6	**Practice** the skills to excel and engage in lifelong learning.	Apply

Faculty signature

ABSTRACT

Arthritis is a disorder that causes swelling, tenderness, inflammation, stiffness etc. in one or more joints. Arthritis is more common in older people and typically worsens with age. While there are many different types of arthritis with different causes and treatments, osteoarthritis is the most prevalent. Osteoarthritis is estimated to affect nearly 237 million people globally, this accounts for almost 3.3% of the human population. Although as of now there is no known cure for arthritis, the benefits of early detection can't be understated.

Knee OA severity is assessed using Kellgren & Lawrence (KL) grades, a five point scale. Previous work on automatically predicting KL grades from radio graph images were based on training shallow classifiers using a variety of hand engineered features. We demonstrate that classification accuracy can be significantly improved using deep convolutional neural network models pre -trained on Image Net and fine-tuned on knee OA images. This leads to the formulation of the prediction of KL grades as classification problem and further improves accuracy. Results on a data set of X-ray images and KL grades from the Osteoarthritis Initiative (OAI) show a sizable improvement over the current state of the art.

The Knee Osteoarthritis Detection helps patients and the doctor to detect Osteoarthritis in patients knees. Along with helping with early detection, this application also detects the severity of the disorder.

INDEX

LIST OF FIGURES

LIST OF TABELS

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

This is a new approach to automatically quantify the severity of knee osteoarthritis (OA) from radio graphs using different convolutional neural networks (CNN). Clinically, knee OA severity is assessed using Kellgren & Lawrence (KL) grades, a five point scale. Previous work on automatically predicting KL grades from radio graph images were based on training shallow classifiers using a variety of hand engineered features. We demonstrate that classification accuracy can be significantly improved using deep convolutional neural network models pre -trained on Image Net and fine-tuned on knee OA images.This leads to the formulation of the prediction of KL grades as a regression problem and further improves accuracy.

This is a fresh method for using X-ray pictures to automatically measure the severity of knee OA. Two stages are required to automatically determine the severity of knee OA: initially, naturally Localizing the knee joints, then categorizing the photos of the localized knee joints.We present a novel method to recognize the knee joints automatically.through the use of a fully convolutional neural network (FCN). We develop convolutional creating from scratch neural networks (CNN) to automatically measure the knee Optimizing. This collaborative training enhances the measurement of knee OA severity overall, with the additional bonus of automatically generating outcomes the multi-class classification.

Osteoarthritis of the knee happens when cartilage in your knee joint breaks down. When this happens, the bones in your knee joint rub together, causing friction that makes your knees hurt, become stiff or swell. Osteoarthritis in the knee can’t be cured but there are treatments that can relieve symptoms and slow your condition’s progress. Any joint in the body can develop osteoarthritis, which is a highly frequent condition. The joints that support the majority of our weight, such the knees and foot, are most likely to be affected. Additionally, frequently used joints, such the hand joints, are frequently impacted. The cartilage, a robust but supple and slippery tissue that covers the surface of the bones in a healthy joint, allows the bones to move freely against one another. Osteoarthritis causes the cartilage in a joint to thin in some places, making the surface rougher. This indicates that the joint doesn't move as easily as it ought to. Knee osteoarthritis (OA) is one major cause of activity limitation and physical disability in older adults. Early detection and intervention can help slow down the OA degeneration. Physicians' grading based on visual inspection I subjective, varied across interpreters, and highly relied on their experience. In our project, we successively apply five convolutional neural networks (CNN) to automatically measure the knee OA severity, as assessed by the Kellgren - Lawrence (KL) grading system.

Firstly, considering the size of knee joints distributed in X-ray images with small variability, we detect knee joints using a customized network. Secondly, we fine-tune the most popular CNN models, including variants to classify the detected knee joint images with a novel adjustable ordinal loss. To be specific, motivated by the ordinal nature of the knee KL grading task, we assign higher penalty to miss classification with larger distance between the predicted KL grade and the real KL grade. The baseline X-ray images from the Osteoarthritis Initiative (OAI) data set are used for evaluation.

1.2 Purpose Of The System

The Knee Osteoarthritis Detection helps doctors and patient to detect Osteoarthritis in their knees. Along with helping with early detection, this application also detects the severity of the disorder.This is a fresh method for using X-ray pictures to automatically measure the severity of knee OA.

1.3 Scope of the System

- Based on the X- rays detecting the knee OA.
- The aim is to apply deep learning in detecting the knee OA and its stage .
- Using various model on the dataset comparing them and stick to the model which got better accuracy.

1.4 Existing System:

In the existing system it become more difficult to accurately predict or detect the severity of knee osteoarthritis.(X-ray) has been traditionally preferred, and remains the main accessible tool for preliminary knee OA diagnosis.

1.5 Proposed System:

This is a fresh method for using X-ray pictures to automatically measure the severity of knee OA. Two stages are required to automatically determine the severity of knee OA: initially, naturally Localizing the knee joints, then categorizing the photos of the localized knee joints using CNN model. Along with traditional CNN model we have also used the various transfer learning methodologies like VGG16,VGG19,resnet50,inception v3 for better accuracy.

CHAPTER 2
REQUIREMENT ANALYSIS

2. REQUIREMENT ANALYSIS

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. Requirements analysis is an important aspect of project management. Requirements can be architectural, structural, behavioral, functional and non-functional.

2.1 Functional Requirements

- Ease of Use: It can be implemented on any platforms.
- High Performance: This is a fresh method for using X-ray pictures to automatically measure the severity of knee OA.As it makes use of different models that help us to make the system high having performance output.
- Maintainability: To minimize life-cycle cost.
- Scalability: To support new requirements and ever increasing scope.
- Cross-Platform Support: To support multiple operating systems to enable maximum selection of available hardware.

2.2 Non-Functional Requirements

A non-functional requirement is a specification that describes the system's operation capabilities and constraints that enhance its functionality. We've covered different types of software requirements, based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system

.

- Software Requirements
- Hardware Requirements
- Usability
- Reliability

- Performance
- Physical Environment
- Resource Requirements

2.2.1 Software Requirements

Software requirements deal with defining software resources requirements and per-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or per-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

- Operating System : Windows
- IDE : Jupiter

2.2.2 Hardware Requirements:

The most common set of requirements defined by any software application is the physical resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list. The following sub sections discuss the various aspects of hardware requirements required for this project.

- Processor : Intel i3 & above
- Ram : Minimum 4GB & above
- Hard Disk : 2GB
- System Type : 64 bit
- Graphic card

2.2.3 Usability:

This is used to Detect the Knee Osteoarthritis helps patients detect the Osteoarthritis in their knees. Along with helping with early detection, this web application also detects the severity of the disorder

2.2.4 Relability:

This system is used to make accuracy between different models, thus it has high. Reliability with value lying between 0 and 1.

2.2.5 Performance:

This is a fresh method for using X-ray pictures to automatically measure the severity of knee OA. Two stages are required to automatically determine the severity of knee OA: initially, naturally Localizing the knee joints, then categorizing the photos of the localized knee joints.As it makes use of different models that help us to make the system high having performance output.

2.2.6 Physical Environment:

It is compatible with any system like a laptop & system.

2.2.7 Resource Requirement:

Required software is to be installed like python idle, jupter notebook, and respected libraries pandas, numpy,sklearn are to be installed in our system to execute our project successfully.

CHAPTER 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Introduction

System analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements on the system. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers.

A detailed study of these processes is made by various techniques like Interviews,reviews, etc. The data collected by these sources is scrutinized to arrive to a conclusion.The conclusion is an understanding of how the system functions. This system is called the existing system. System analysis is the process of compiling and analyzing data, diagnosing issues, and using the data to suggest system changes. System analysis is a problem- solving process that necessitates frequent contact between system developers and users. Any system development process should start with a system analysis phase. The system is meticulously examined and analyzed. The system analyst acts as an interrogator and delves deeply into how the current system functions. The inputs to the system are recognized and the system is seen as a whole.

System analysis is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of an interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the inputs to the system are identified. The outputs from the organization are traced through the various processing that the inputs phase through in the organization.

The organization's outputs can be followed through the numerous processing steps that its inputs go through.These procedures are thoroughly studied using a variety of methodologies, including interviews, reviews, etc. To come to a conclusion, the information gathered from these sources is carefully examined. Understanding how the system works is the conclusion. The current system is the name of this system.

UML Diagrams

- UML is a notation that resulted from the unification of Object Modeling technique and Object Oriented Software Technology.
- Unified Modeling Language is the language used to visualize,specify,construct and document any component of software engineering.
- The software engineer can use the modelling notation that is governed by a set of syntactic, semantic, and pragmatic rules to express an analytical model using the Unified Modeling Language.
- System analysis is the most innovative and difficult stage in the life cycle. A final system and the method by which it is created are both described by the term analysis. It alludes to the technical requirements that will be used when the candidate system is implemented.

3.2 Usecase Diagram

The **Usecase diagram** is a graphic that is used to define the core elements and processes that make up a system. The key elements are termed as "*actors*" and the processes are called "actions". It shows which actors interact with each usecase. Usecase diagrams are created to visualize the relationship between actors and usecases. A usecase is a pattern of behaviour the system exhibits. Each usecase is a sequence of related transactions performed by an actor and the system in a dialogue.
A flow of events document is created for each usecases.

Use case diagram is created to visualize the relationships between actors and use cases. A use case is a pattern of behaviour the system exhibits. Each use case is a sequence of related transactions performed by an actor and the system. Diagrammatically actor and use case are represented by stick figure and oval respectively. The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

- It gathers the system's needs.
- It depicts the external view of the system.
- It recognizes the internal as well as external factors that influence the system.
- It represents the interaction between the actors.

Symbols

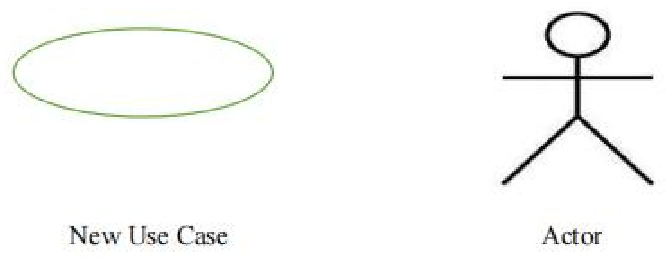


Fig:3.2.1 Symbols In Usecase Diagram

Relationships

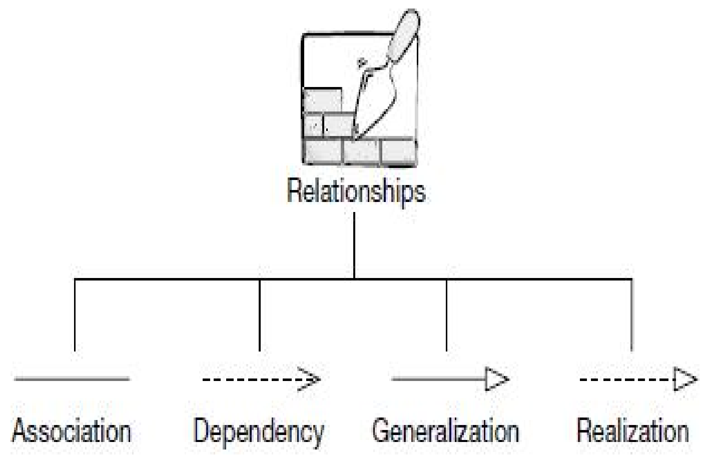


Fig:3.2.2 Relationships

A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UML is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modeled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself.

- Use cases: Horizontally shaped ovals that represent the different uses that a user might have.
- Actors: Stick figures that represent the people actually employing the use cases.
- Associations: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- System boundary boxes: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
- Packages: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.
- Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
- System: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
- Goals: The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

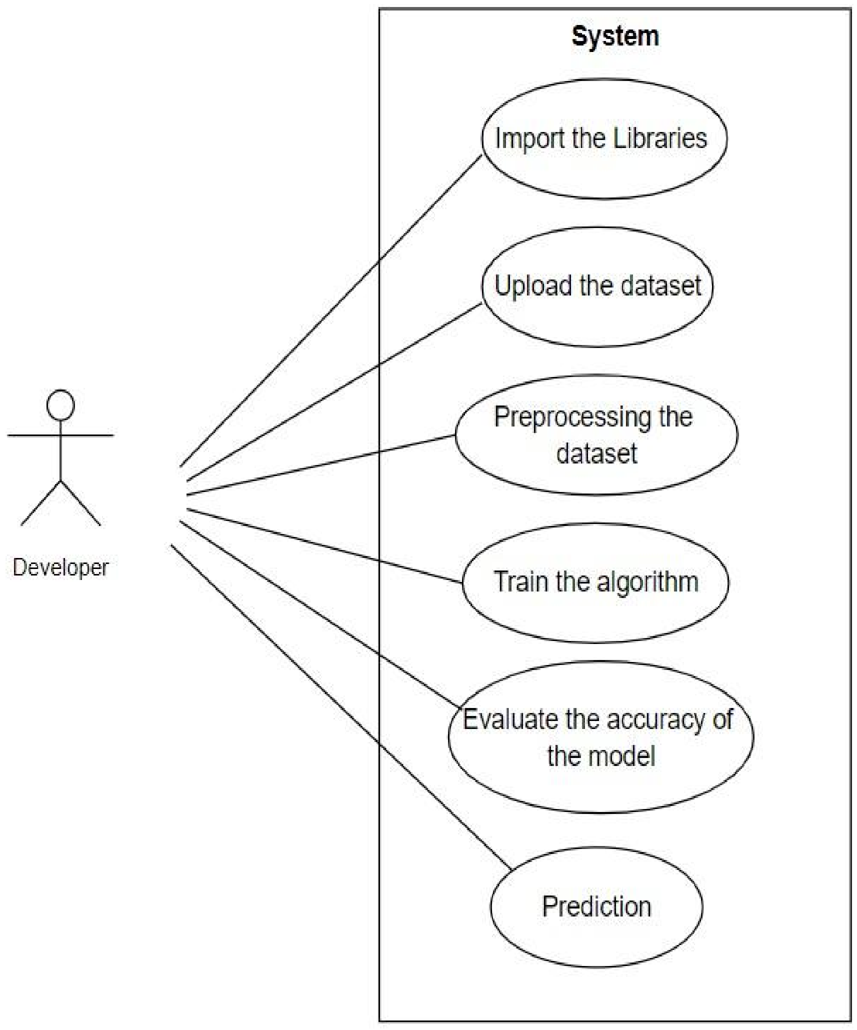


Fig: 3.2.3 Usecase Diagram

Following below are the breakdown of the individual Use Cases. Each showing the Actors, Input and Output respectively:

ACTORS AND THEIR USE CASES

ACTOR: DEVELOPER

ACTIONS:

- **Import Libraries:** Before importing, the libraries need to be installed in our computer using pip or conda commands.Later, import them using keyword import in python.

- **Upload Dataset:** The dataset is uploaded from local location to jupyter editor and its path is assigned to a variable.
- **Preprocessing Of Dataset:** The dataset consisting of images of different size they need to be re sized and conversion of channels also been done in this phase.
- **Train Algorithm:** The preprocessed dataset is given to the algorithm and model is evaluated with epochs.
- **Evaluating Of Model:** On evaluation of algorithm we can get the accuracy and loss of both trained data and test data.
- **Prediction:** It predicts the new image and classifies the severity based upon the trained data after reshaping and re sizing been done.

CHAPTER 4
SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Introduction

An efficient system development life cycle (SDLC) should produce a high-quality system that meets customer expectations, is completed in accordance with time and cost estimates, and functions effectively and efficiently in the information technology infrastructure that is currently in place and that is anticipated to be built in the future. A conceptual model called the System Development Life Cycle (SDLC) outlines guidelines and processes for creating and modifying systems over the course of their entire lives.

System design is the process of defining the architecture, models, interfaces,and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of system analysis, system architecture and system engineering.

The design phase describes how the system will fulfil the user requirements. To achieve this, we must create both logical and physical design. In this phase the system design functions and operations are described.System design is transition from a user oriented, document oriented to programmers.The design is a solution, a “how to” approach to the creation of a new system.

4.2 System Architecture

A system architecture, sometimes known as a systems architecture, is a conceptual model that describes a system's behaviour, structure, and other aspects. A description and representation of a system's architecture that is set up to facilitate analysis of a system's behaviour and structure.

The knee osteoarthritis detection and severity using deep learning architectural design is a diagram that shows the connections between the main software structural components, the design patterns that can be used to meet the system requirements, and the limitations that can affect how architectural design patterns can be used.

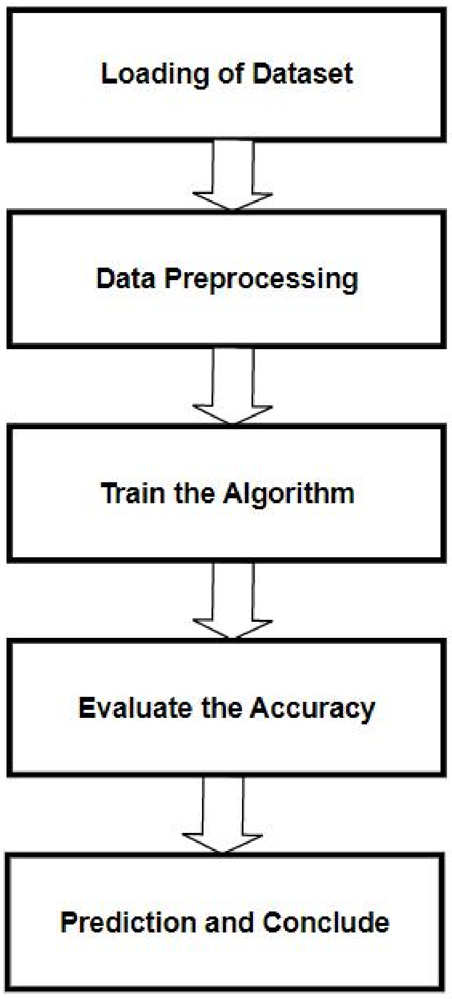


Fig: 4.2 System Architecture

4.3 System Object Model

4.3.1 Introduction

IBM created the System Object Model (SOM), an object-oriented library packaging system that enables different programming languages to share class libraries regardless of the language in which they were initially written.When class libraries are shared between object-oriented and non-object-oriented languages, several interoperability and reuse issues arise.

The system object model was designed to address these issues.SOM was created to be usable with both mainframe and desktop computers from IBM. It acts as a differentiating object-oriented model from others seen in object-oriented programming languages. System object model was intended to be used as a solution to many of the interoperability and reuse problems that occur while sharing class libraries between object-oriented and non-object-oriented languages.

4.4 Object Description

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of a system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

4.4.1 Object

Object is any entity that can be manipulated by the commands of programming languages such as value, variable, function, or data structure.

4.4.2 Class Diagram

- A Class diagrams describe the static structure of a system, or how it is structured rather than how it behaves.
- Class diagram gives an overview of a system by showing its classes and the relationships among them.
- UML class is a rectangle divided into: class name, attributes, and operations.
- Our class diagram has three kinds of relationships.
  - Association
  - Aggregation
  - Generalization.

A class diagram showing the systems classes, their attributes, operations, and collaborations and the relationships among objects. The class diagrams is the main building lock of object oriented modelling. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application and the classes to be programmed.

It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system . The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with objectoriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as :

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

Class diagrams are the most popular UML diagrams used for construction of software applications. It is very important to learn the drawing procedure of class diagram.

Class diagrams have a lot of properties to consider while drawing but here the diagram will be considered from a top level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represent the whole system.

The following points should be remembered while drawing a class diagram −

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. at the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct

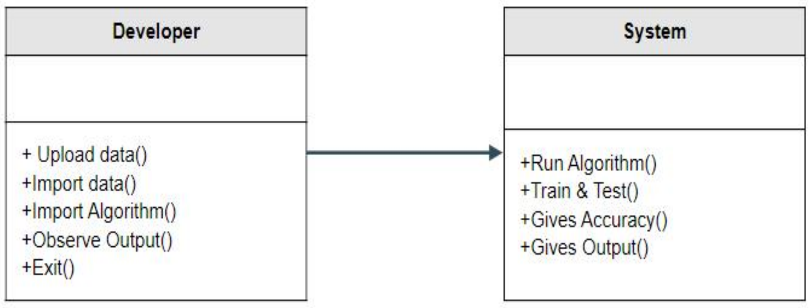


Fig: 4.4.2 Class Diagram

4.5 Dynamic Model

4.5.1 Activity Diagram

An activity diagram is another important diagram in UML to describe the dynamic aspects of the system. An activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. The basic purposes of activity diagrams are similar to the other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but the activity diagram is used to show message flow from one activity to another. State Chart diagram is one of the five UML diagrams used to model the dynamic nature of a system.They define different states of an object during its lifetime and these states are changed by events.

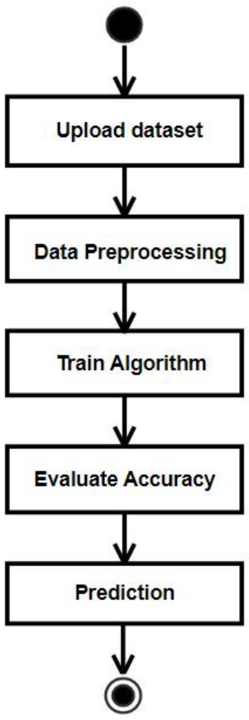


Fig: 4.5.1 Activity Diagram

4.5.2 State Chart Diagram

State Chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events. State Chart diagram describes the flow of control from one state to another state.

States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State Chart diagram is to model lifetime of an object from creation to termination.

State Chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.
Following are the main purposes of using State Chart diagrams

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object

State Chart diagram defines the states of a component and these state changes are dynamic in nature. Its specific purpose is to define the state changes triggered by events.

Events are internal or external factors influencing the system. State Chart diagrams are used to model the states and also the events operating on the system. When implementing a system, it is very important to clarify different states of an object during its life time and State Chart diagrams are used for this purpose. When these states and events are identified, they are used to model it and these models are used during the implementation of the system. If we look into the practical implementation of State Chart diagram, then it is mainly used to analyze the object states influenced by events. This analysis is helpful to understand the system behavior during its execution.
The main usage can be described as

- To model the object states of a system.
- To model the reactive system. Reactive system consists of reactive objects.
- To identify the events responsible for state changes.
- Forward and reverse engineering

State Chart diagram is used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.

State Chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs. Before drawing a State Chart diagram we should clarify the following points

- Identify the important objects to be analyzed.
- Identify the states.
- Identify the events.

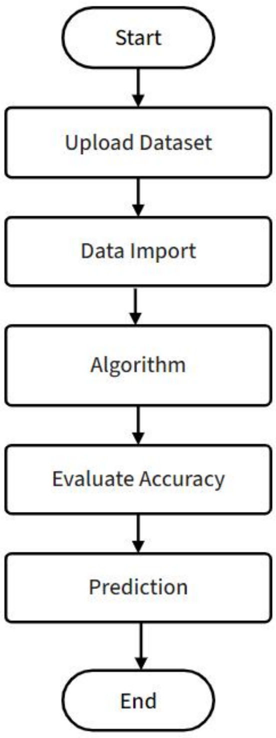


Fig: 4.5.2 State Chart Diagram

4.5.3 Sequence diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. A sequence diagram shows object interactions arranged in time sequence. The interaction that takes place in collaboration that either realizes a use case or an operation. Sequence diagrams provide high level interactions between user of the system and the system, between the system and the other systems, or between subsystems. Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.

- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

Sequence Diagrams are graphs of relationships which describe how activities are done.Here, the system interacts with the patient. It takes the necessary data from the user/patient, which supplies the system the necessary queries and then sends the output of the generated system.

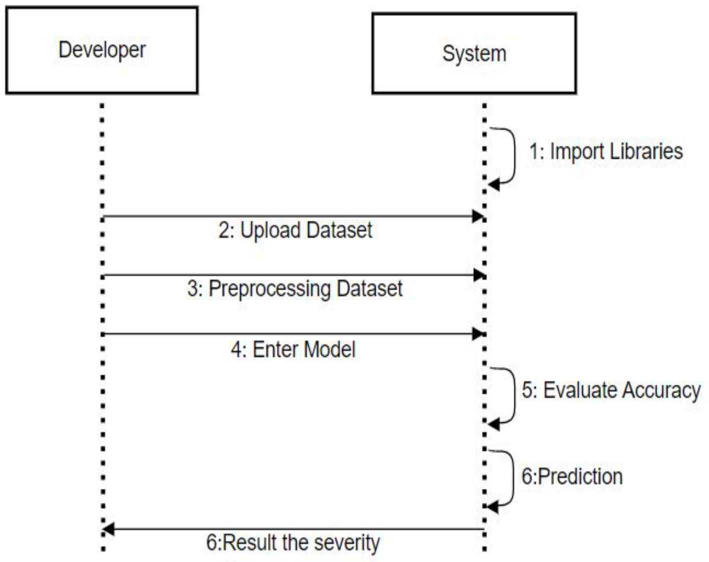


Fig:4.5.3 Sequence Diagram

4.5.4 Communication Diagram

The communication diagram and the sequence diagram are similar. They're semantically equivalent, that is, they present the same information, and you can turn a communication to a sequence diagram and vice versa. The main distinction between them is that the communication diagram arranges elements according to space, the sequence diagram is according to time.

Communication diagrams offer benefits similar to sequence diagrams, but they will offer a better understanding of how components communicate and interact with each other rather than solely emphasizing the sequence of events. They can be a useful reference for businesses, organizations, and engineers who need to visualize and understand the physical communications within a program. Try drawing a sequence diagram to:

- Model the logic of a sophisticated procedure, function, or operation.
- Identify how commands are sent and received between objects or components of a process.
- Visualize the consequences of specific interactions between various components in a process.
- Plan and understand the detailed functionality of an existing or future scenario.

A communication diagram offers the same information as a sequence diagram, but while a sequence diagram emphasizes the time and order of events, a communication diagram emphasizes the messages exchanged between objects in an application. Sequence diagrams can fall short of offering the "big picture."

This is where communication diagrams come in and offer that broader perspective within a process.

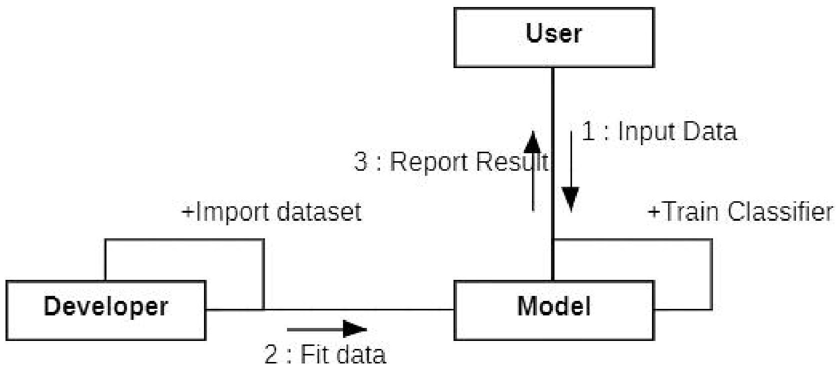


Fig:4.5.4 Communication Diagram

4.6 Static Model

4.6.1 Component Diagram

UML Component diagrams are used in modelling the physical aspects of object oriented systems that are used for visualizing, specifying, and documenting componentbased systems and also for constructing executable systems through forward and reverse engineering.

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system.

Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole. The purpose of the component diagram can be summarized as −

- Visualize the components of a system.
- Construct executable by using forward and reverse engineering.
- Describe the organization and relationships of the components.

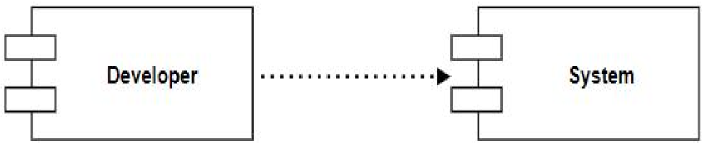


Fig: 4.6.1 Component Diagram

4.6.2 Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

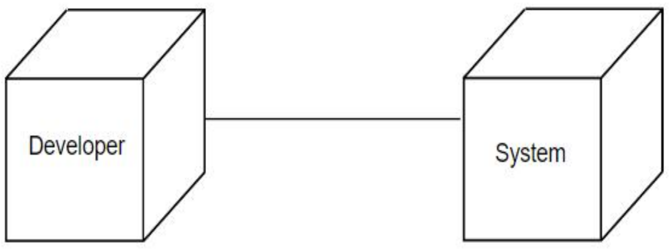


Fig:4.6.2 Deployment Diagram

CHAPTER 5
IMPLEMENTATION

5. IMPLEMENTATION

5.1 Software Used

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Jupyter notebook is one of the most powerful IDE that works with all major programming languages like Python , Java versions R, Julia, and Scala. It is maintained by project jupyter . This IDE is loaded with rich features and functionalities that, one can possibly imagine.

Jupyter Notebook benefits

Originally developed for data science applications written in Python, R, and Julia, Jupyter Notebook is useful in all kinds of ways for all kinds of projects:

Data visualizations: Most people have their first exposure to Jupyter Notebook by way of a data visualization, a shared notebook that includes a rendering of some data set as a graphic. Jupyter Notebook lets you author visualizations, but also share them and allow interactive changes to the shared code and data set.

Code sharing: Cloud services like GitHub and Pastebin provide ways to share code, but they're largely non-interactive. With a Jupyter Notebook, you can view code, execute it, and display the results directly in your web browser.

Live interactions with code: Jupyter Notebook code isn't static; it can be edited and re-run incrementally in real time, with feedback provided directly in the browser. Notebooks can also embed user controls (e.g., sliders or text input fields) that can be used as input sources for code.

Documenting code samples: If you have a piece of code and you want to explain line-by-line how it works, with live feedback all along the way, you could embed it in a Jupyter Notebook. Best of all, the code will remain fully functional you can add interactivity along with the explanation, showing and telling at the same time.

Python Programming

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

There are two major Python versions: Python2 and Python3

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

The biggest strength of Python is huge collection of standard library which can be used for the following :

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

There are more advantages of using Python Programming Language.
Some of them are:

- **Extensive Libraries**

  Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unittesting, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don‘t have to write the complete code for that manually.
- **Extensible**

  As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.
- **Embeddable**

  Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.
- **Improved Productivity**

  The language‘s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

- **IOT Opportunities**

  Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

- **Simple and Easy**

  When working with Java, you may have to create a class to print „Hello World‟. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

- **Readable**

  Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

- **Object-Oriented**

  This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

- **Free and Open-Source**

  Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

- **Portable**

  When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn‘t the same with Python. Here, you need to code. only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

- **Interpreted**

  Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages

Advantages of Python Over Other Languages

- **Less Coding**

  Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don‘t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.
- **Affordable**

  Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.
- **Python is for Everyone**

  Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we‘ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let‘s now see the downsides of choosing Python over another language.

- **Speed Limitations**

  We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn‘t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.
- **Weak in Mobile Computing and Browsers**

  While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.
- **Design Restrictions**

  As you know, Python is dynamically-typed. This means that you don‘t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what‘s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise runtime errors.
- **Underdeveloped Database Access Layers**

  Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python‘s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

This was all about the Advantages and Disadvantages of Python Programming Language.

Deep Learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain albeit far from matching its ability allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

Deep learning vs. machine learning

If deep learning is a subset of machine learning, how do they differ? Deep learning distinguishes itself from classical machine learning by the type of data that it works with and the methods in which it learns.

Machine learning algorithms leverage structured, labeled data to make predictions meaning that specific features are defined from the input data for the model and organized into tables. This doesn't necessarily mean that it doesn't use unstructured data; it just means that if it does, it generally goes through some pre-processing to organize it into a structured format.Deep learning eliminates some of data pre-processing that is typically involved with machine learning. These algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts. For example, let's say that we had a set of photos of different pets, and we wanted to categorize by "cat", "dog", "hamster", etc. Deep learning algorithms can determine which features (e.g. ears) are most important to distinguish each animal from another. In machine learning, this hierarchy of features is established manually by a human expert.

Then, through the processes of gradient descent and back propagation, the deep learning algorithm adjusts and fits itself for accuracy, allowing it to make predictions about a new photo of an animal with increased precision.

Machine learning and deep learning models are capable of different types of learning as well, which are usually categorized as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning utilizes labeled datasets to categorize or make predictions; this requires some kind of human intervention to label input data correctly. In contrast, unsupervised learning doesn't require labeled datasets, and instead, it detects patterns in the data, clustering them by any distinguishing characteristics. Reinforcement learning is a process in which a model learns to become more accurate for performing an action in an environment based on feedback in order to maximize the reward.

How deep learning works?

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

Another process called back propagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and back propagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

Applications Of Deep Learning

- Automated Driving: Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.
- Aerospace and Defense: Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.
- Medical Research: Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.
- Industrial Automation: Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.
- Electronics: Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

5.2 Algorithms

Convolutional Neural Network(CNN)

A convolutional neural network (CNN) typically comprises multiple convolutional and sub-sampling layers, optionally followed by fully-connected layers like a standard multi-layer neural network. A CNN exploits the 2D spatial structure images to learn translation invariant features. The main advantage of CNN over fully connected networks is that they are easier to train and have fewer parameters with the same number of hidden units. In this work, we train CNNs from scratch to automatically quantify knee OA severity using

X-ray images. This involves two main steps: (1) automatically detecting and extracting the region of interest (ROI) and localizing the knee joints, (2) classifying the localized knee joints. We introduce a fully-convolutional neural network (FCN) based method to automatically localize the knee joints. A FCN is an end-to-end network trained to make pixel-wise predictions. Our FCN based method is highly accurate for localizing knee joints and the FCN can easily fit into an end-to-end network trained to quantify knee OA severity.

VGG16

VGG16 refers to the VGG model, also called VGGNet. It is a convolution neural network (CNN) model supporting 16 layers. K. Simonyan and A. Zisserman from Oxford University proposed this model and published it in a paper called Very Deep Convolutional Networks for Large-Scale Image Recognition.A VGG network consists of small convolution filters. VGG16 has three fully connected layers and 13 convolutional layers.
Here is a quick outline of the VGG architecture:

- **Input** VGGNet receives a 224×224 image input. In the ImageNet competition, the model's creators kept the image input size constant by cropping a 224×224 section from the center of each image.
- **Convolutional layers** the convolutional filters of VGG use the smallest possible receptive field of 3×3. VGG also uses a 1×1 convolution filter as the input's linear transformation.
- **ReLu activation** next is the Rectified Linear Unit Activation Function (ReLU) component, AlexNet's major innovation for reducing training time. ReLU is a linear function that provides a matching output for positive inputs and outputs zero for negative inputs. VGG has a set convolution stride of 1 pixel to preserve the spatial resolution after convolution (the stride value reflects how many pixels the filter "moves" to cover the entire space of the image).

- **Hidden layers** all the VGG network's hidden layers use ReLU instead of Local Response Normalization like AlexNet. The latter increases training time and memory consumption with little improvement to overall accuracy.
- **Pooling layers** A pooling layer follows several convolutional layers—this helps reduce the dimensionality and the number of parameters of the feature maps created by each convolution step. Pooling is crucial given the rapid growth of the number of available filters from 64 to 128, 256, and eventually 512 in the final layers.
- **Fully connected layers** VGGNet includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

VGG19

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The“deep” refers to the number of layers with VGG-19 consisting of 19 convolutional layers.Transfer learning provides a very effective idea to solve the above problems of deep learning. It uses a pretrained deep convolutional neural network that has been trained on another dataset**.** VGG19 has three fully connected layers and 16 convolutional layers.
Here is a quick outline of the VGG architecture:

- **Input** VGGNet receives a 224×224 image input. In the ImageNet competition, the model's creators kept the image input size constant by cropping a 224×224 section from the center of each image.
- **Convolutional layers** the convolutional filters of VGG use the smallest possible receptive field of 3×3. VGG also uses a 1×1 convolution filter as the input's linear transformation.
- **ReLu activation** next is the Rectified Linear Unit Activation Function (ReLU) component, AlexNet's major innovation for reducing training time. ReLU is a linear function that provides a matching output for positive inputs and outputs zero for negative inputs. VGG has a set

convolution stride of 1 pixel to preserve the spatial resolution after convolution (the stride value reflects how many pixels the filter "moves" to cover the entire space of the image).

- **Hidden layers** all the VGG network's hidden layers use ReLU instead of Local Response Normalization like AlexNet. The latter increases training time and memory consumption with little improvement to overall accuracy.
- **Pooling layers** A pooling layer follows several convolutional layers—this helps reduce the dimensionality and the number of parameters of the feature maps created by each convolution step. Pooling is crucial given the rapid growth of the number of available filters from 64 to 128, 256, and eventually 512 in the final layers.
- **Fully connected layers** VGGNet includes three fully connected layers. The first two layers each have 4096 channels, and the third layer has 1000 channels, one for every class.

Resnet50

ResNet stands for Residual Network and is a specific type of convolutional neural network (CNN) introduced in the 2015 paper "Deep Residual Learning for Image Recognition" by He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. CNNs are commonly used to power computer vision applications.

ResNet-50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer). Residual neural networks are a type of artificial neural network (ANN) that forms networks by stacking residual blocks.The ResNet architecture follows two basic design rules. First, the number of filters in each layer is the same depending on the size of the output feature map. Second, if the feature map's size is halved, it has double the number of filters to maintain the time complexity of each layer.

ResNet-50 has an architecture based on the model depicted above, but with one important difference. The 50-layer ResNet uses a bottleneck design for the building block. A bottleneck residual block uses 1×1 convolutions, known as a "bottleneck", which reduces the number of parameters and matrix multiplications. This enables much faster training of each layer. It uses a stack of three layers rather than two layers.

The 50-layer ResNet architecture includes the following elements, as shown in the table below:

- **A 7×7 kernel convolution** alongside 64 other kernels with a 2-sized stride.
- **A max pooling layer** with a 2-sized stride.
- **9 more layers**—3×3,64 kernel convolution, another with 1×1,64 kernels, and a third with 1×1,256 kernels. These 3 layers are repeated 3 times.
- **12 more layers** with 1×1,128 kernels, 3×3,128 kernels, and 1×1,512 kernels, iterated 4 times.
- **18 more layers** with 1×1,256 cores, and 2 cores 3×3,256 and 1×1,1024, iterated 6 times.
- **9 more layers** with 1×1,512 cores, 3×3,512 cores, and 1×1,2048 cores iterated 3 times.(up to this point the network has 50 layers)
- **Average pooling**, followed by a fully connected layer with 1000 nodes, using the softmax activation function.

Inception v3

Inception v3 is a convolutional neural network for assisting in image analysis and object detection, and got its start as a module for GoogLeNet. It is the third edition of Google's Inception Convolutional Neural Network, originally introduced during the ImageNet Recognition Challenge. The design of Inceptionv3 was intended to allow deeper networks while also keeping the number of parameters from growing too large: it has "under 25 million parameters", compared against 60 million for AlexNet.

Just as ImageNet can be thought of as a database of classified visual objects, Inception helps classification of objects in the world of computer vision. The Inceptionv3 architecture has been reused in many different applications, often used "pre-trained" from ImageNet. One such use is in life sciences.Inception v3 is an image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using Softmax.

5.3 Transfer learning

The reuse of a pre-trained model on a new problem is known as transfer learning in machine learning. A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning. You could, for example, use the information gained during training to distinguish beverages when training a classifier to predict whether an image contains cuisine.

The knowledge of an already trained machine learning model is transferred to a different but closely linked problem throughout transfer learning. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the model’s training knowledge to identify other objects such as sunglasses.

How Transfer Learning Works

In computer vision, neural networks typically aim to detect edges in the first layer, forms in the middle layer, and task-specific features in the latter layers. The early and central layers are employed in transfer learning, and the latter layers are only retrained. It makes use of the labelled data from the task it was trained on. Transfer learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task. To put it simply—a model trained on one task is repurposed on a second, related task as an optimization that allows rapid progress when modeling the second task.

By applying transfer learning to a new task, one can achieve significantly higher performance than training with only a small amount of data. Transfer learning is so common that it is rare to train a model for an image or natural language processing-related tasks from scratch. Instead, researchers and data scientists prefer to start from a pre-trained model that already knows how to classify objects and has learned general features like edges, shapes in images.

ImageNet, AlexNet, and Inception are typical examples of models that have the basis of Transfer learning.

Traditional Machine Learning vs.Transfer Learning

Deep learning experts introduced transfer learning to overcome the limitations of traditional machine learning models.

Let's have a look at the differences between the two types of learning.

- Traditional machine learning models require training from scratch, which is computationally expensive and requires a large amount of data to achieve high performance. On the other hand, transfer learning is computationally efficient and helps achieve better results using a small data set.
- Traditional ML has an isolated training approach where each model is independently trained for a specific purpose, without any dependency on past knowledge. Contrary

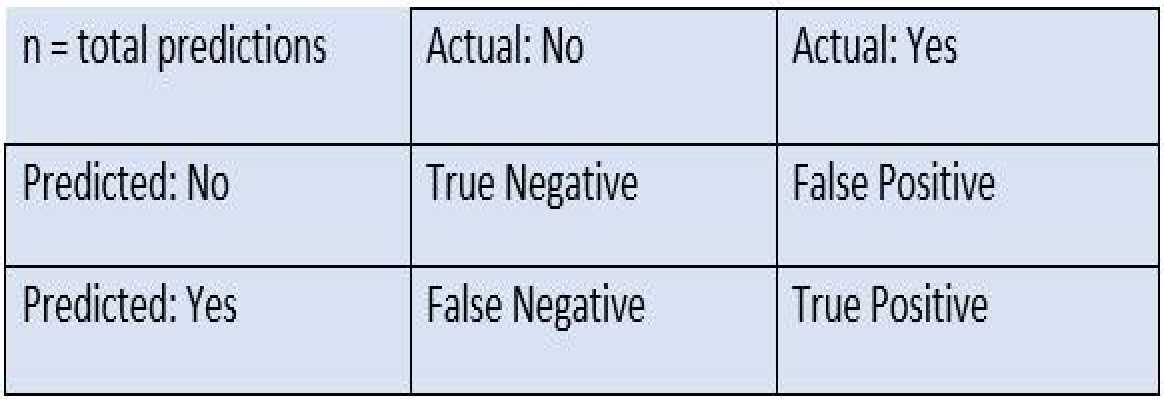
to that, transfer learning uses knowledge acquired from the pre-trained model to proceed with the task. To paint a better picture of it: One can not use the pre-trained model of ImageNet with biomedical images because ImageNet does not contain images belonging to the biomedical field.

- Transfer learning models achieve optimal performance faster than the traditional ML models. It is because the models that leverage knowledge (features, weights, etc.) from previously trained models already understand the features. It makes it faster than training neural networks from scratch.

5.4 Confusion Matrix

A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.

- The matrix is divided into two dimensions, that are predicted values and actual values along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
- It looks like the below table:



n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

Fig: 5.4 confusion matrix

The above table has the following cases:

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

From the confusion matrix, we can find the following metrics

- **Accuracy:** Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Precision:** Precision is a measure of how accurate a model's positive predictions are. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall:** Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It is the ratio of the number of true positive (TP) instances to the sum of true positive and false negative (FN) instances.

$$\text{Recall} = \frac{TP}{TP+FN}$$

F1-Score: F1-score is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

$$\text{F1-Score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

5.5 Source Code

```
#importing of modules
import cv2,os
import numpy as np
#keras module
import keras
from keras.models import Sequential
from keras.layers import Dense,Activation,Flatten,Dropout
from keras.layers import Conv2D,MaxPooling2D
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils
#matplotlib module
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
#sklearn module
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix


data_path=(r'C:\Users\Navya\OneDrive\Desktop\PYTHON PRGMS\Knee X-ray Images\MedicalExpert-I')
print(data_path)
```

```
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]


label_dict=dict(zip(categories,labels)) #empty dictionary
print(label_dict)
print(categories)
print(labels)


img_size=224
data=[]
label=[]
for category in categories:
   folder_path=os.path.join(data_path,category)
   img_names=os.listdir(folder_path)
   for img_name in img_names:
      img_path=os.path.join(folder_path,img_name)
      img=cv2.imread(img_path)
      try:
         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         resized=cv2.resize(gray,(img_size,img_size))
         #resizing the image  into 224 x 224, since we need a fixed common size for all
the images in the dataset
         data.append(resized)
         label.append(label_dict[category])
         #appending the image and the label(categorized) into the list (dataset)
      except Exception as e:
         print('Exception:',e)
         #if any exception rasied, the exception will be printed here. And pass to the
next image
```

```
data=np.array(data)/255.0
data=np.reshape(data,(data.shape[0],img_size,img_size,1))
label=np.array(label)
from keras.utils import np_utils
new_label=np_utils.to_categorical(label)

import tensorflow as tf
# Resize the images to (224, 224)
x_train_resized = tf.image.resize(data, (224, 224))
x_test_resized = tf.image.resize(data, (224, 224))

# Convert the images to RGB color format with 3 channels
x_train_resized = tf.image.grayscale_to_rgb(x_train_resized)
x_test_resized = tf.image.grayscale_to_rgb(x_test_resized)

# Normalize the pixel values to be in the range [0, 1]
x_train_resized = x_train_resized / 255.0
x_test_resized = x_test_resized / 255.0



from PIL import Image
import os
# specify the path to the directory containing the images
directory =(r'C:\Users\Navya\OneDrive\Desktop\PYTHON PRGMS\Knee X-ray
Images\MedicalExpert-I')

# specify the new size for the images
new_size = (224, 224)

# use a for loop to iterate over all the files in the directory
```

```
for filename in os.listdir(directory):
    # check if the file is an image
    if filename.endswith('.jpg') or filename.endswith('.png'):
        # open the image using the Image.open() function
        image = Image.open(os.path.join(directory, filename))
        # resize the image
        image = image.resize(new_size)
        # save the resized image
        image.save(os.path.join(directory, filename))


from PIL import Image
import os


# specify the path to the directory containing the images
directory =(r'C:\Users\Navya\OneDrive\Desktop\PYTHON PRGMS\Knee X-ray
Images\MedicalExpert-II')


# specify the new size for the images
new_size = (224, 224)


# use a for loop to iterate over all the files in the directory
for filename in os.listdir(directory):
    # check if the file is an image
    if filename.endswith('.jpg') or filename.endswith('.png'):
        # open the image using the Image.open() function
        image = Image.open(os.path.join(directory, filename))
        # resize the image
        image = image.resize(new_size)
        # save the resized image
        image.save(os.path.join(directory, filename))
```

```
from keras.applications import VGG19
from keras.layers import Dense, Flatten
from keras.models import Model
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
# Load the VGG19 model
base_model = VGG19(weights='imagenet', include_top=False, input_shape=(224,
224, 3))

# Freeze the layers of the VGG19 model
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top of the VGG19 model
x = base_model.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(5, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# Compile the model
model.compile(optimizer=Adam(lr=1e-4),loss='categorical_crossentropy',
metrics=['accuracy'])

# Define the data generators for train and test sets
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
```

```
test_datagen = ImageDataGenerator(rescale=1./255)


# Load the data

train_data=train_datagen.flow_from_directory(directory=(r'C:\Users\Navya\OneDrive
\Desktop\PYTHON PRGMS\Knee X-ray Images\MedicalExpert-I'),

                                            target_size=(224, 224),

                                            batch_size=32,

                                            class_mode='categorical')


test_data=test_datagen.flow_from_directory(directory=(r'C:\Users\Navya\OneDrive\
Desktop\PYTHON PRGMS\Knee X-ray Images\MedicalExpert-II'),

                                           target_size=(224, 224),

                                           batch_size=32,

                                           class_mode='categorical')



history=model.fit(train_data,epochs=20, validation_data=test_data)


model.save('model.h5')


from matplotlib import pyplot as plt

# plot the training loss and accuracy

N = 20 #number of epochs

plt.style.use("ggplot")

plt.figure()

plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")

plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")

plt.plot(np.arange(0, N), history.history["accuracy"], label="train_acc")

plt.plot(np.arange(0, N), history.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="center right")
plt.savefig("CNN_Model")

print("Train accuracy:", history.history['accuracy'][-1])
print("Train loss:", history.history['loss'][-1])
print("Test accuracy:", history.history['val_accuracy'][-1])
print("Test loss:", history.history['val_loss'][-1])

from keras.preprocessing import image
import numpy as np
from PIL import Image

# Load an image to classify
img_path = (r'C:\Users\Navya\OneDrive\Desktop\PYTHON PRGMS\Knee X-ray
Images\MedicalExpert-III\ModerateG3 (1).png')
img = Image.open(img_path).resize((224, 224))
x = np.array(img)
x = np.expand_dims(x, axis=0)
x = x / 255.0

# Make a prediction
prediction = model.predict(x)
predicted_class = np.argmax(prediction)
class_names = ['class_1', 'class_2', 'class_3', 'class_4', 'class_5']
print("Predicted class:", class_names[predicted_class])

from sklearn.metrics import confusion_matrix
```

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Make predictions on test set
y_pred = model.predict(test_data)
y_pred = np.argmax(y_pred, axis=1) # Convert probabilities to class index

# Get true labels
y_true = test_data.classes

# Get class names
class_names = list(test_data.class_indices.keys())

# Calculate confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Create a heatmap from the confusion matrix
sns.heatmap(cm, annot=True, fmt="d", cmap="YlGnBu", xticklabels=class_names,
yticklabels=class_names)

# Add labels and title
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title('Confusion Matrix')

# Show the plot
plt.show()
```

CHAPTER 6

TESTING

6. TESTING

6.1 Introduction

A test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some setting, an oracle could be a requirement or use case. It may take a test case to determine that a software program or system is functioning correctly. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites.

6.1.1 Levels of Testing

In order to uncover the errors present in different phases, we have the concept of levelsof testing. The basic levels of testing are

- Client needs Acceptance
- Testing System Testing
- Design Integration
- Testing
- Code Unit Testing

6.1.2 Software Testing Strategies

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

Unit Testing

Unit Testing is a level of software testing where individual units/ Components of the software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

Integration Testing

Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in integration Testing. Integration Testing is the second level of testing performed after Unit Testing and before System Testing.

Validation Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed in an appropriate environment.

System Testing

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. System testing is performed on the entire system in the context of either functional requirement specifications (FRS) or system requirement specification (SRS), or both. System testing tests not only the design but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software or hardware requirements specification.

6.2 Test Cases

Test Case ID	**Test Case Description**	**Test Data**	**Expected Outcome**	**Actual Outcome**	**Pass/Fail**
TC001	Detection of knee osteoarthritis	A set of knee X-ray images with varying degrees of osteoarthritis	The model accurately identifies the presence or absence of osteoarthritis in the images	If there is no osteoarthritis in images then it classified the image into normal degree i.e [class_1]	Pass
TC002	Severity classification of knee osteoarthritis	A set of knee X-ray images with varying degrees of osteoarthritis, along with corresponding severity labels	The model accurately classifies the severity of osteoarthritis in the images	If model predicts the osteoarthritis then it classified the severity into its corresponding degree i.e [class_2,class_3, class_4,class_5]	Pass

Fig: 6.2 Test cases for developer

.

CHAPTER 7
OUTPUT SCREENS

7. OUTPUT SCREENS

7.1 CNN Model

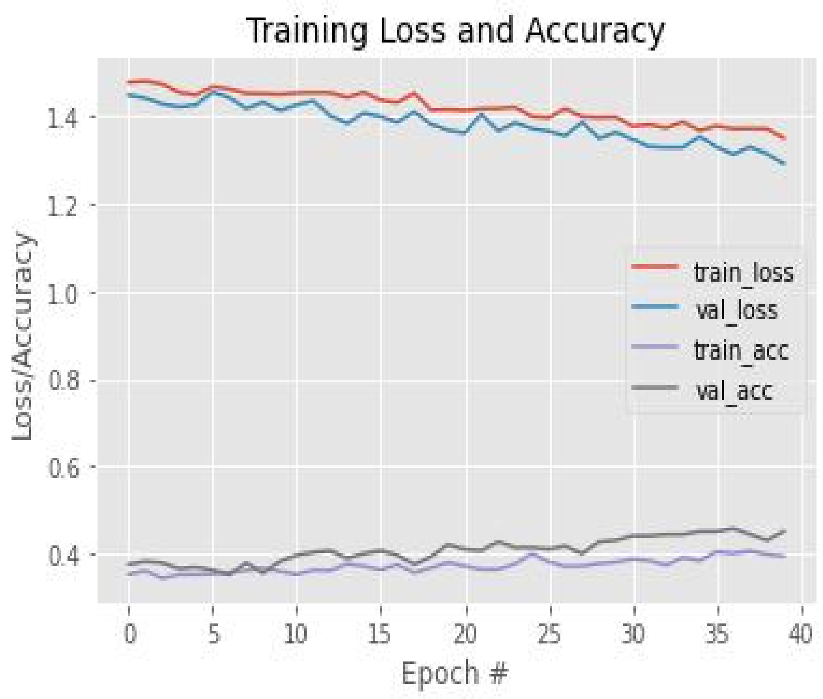


Fig: 7.1.1 Training ,validation accuracy and loss

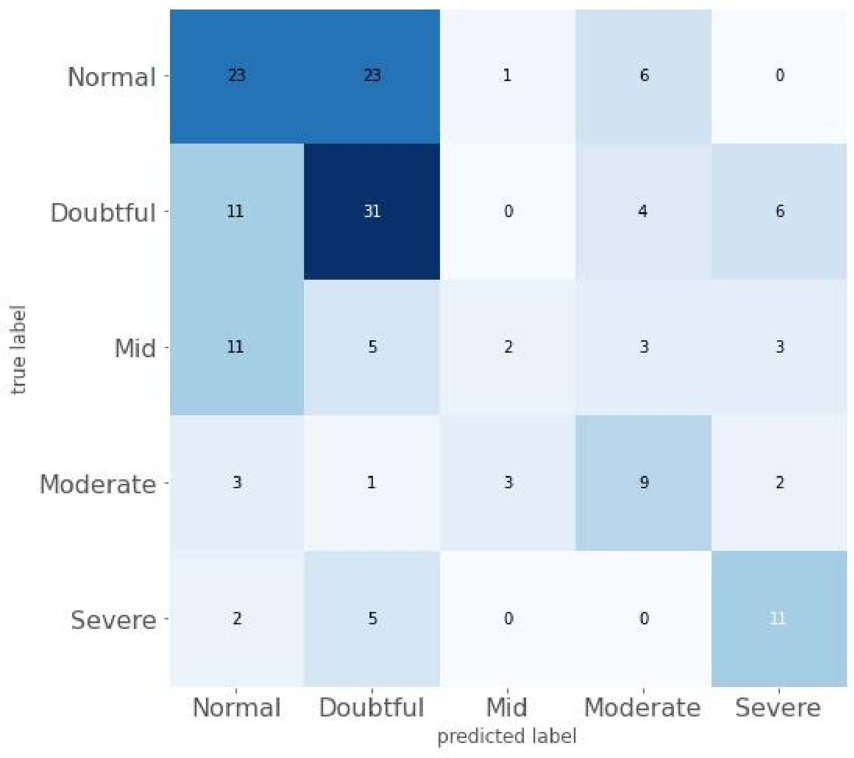


Fig: 7.1.2 Confusion matrix for CNN model

7.2 VGG19

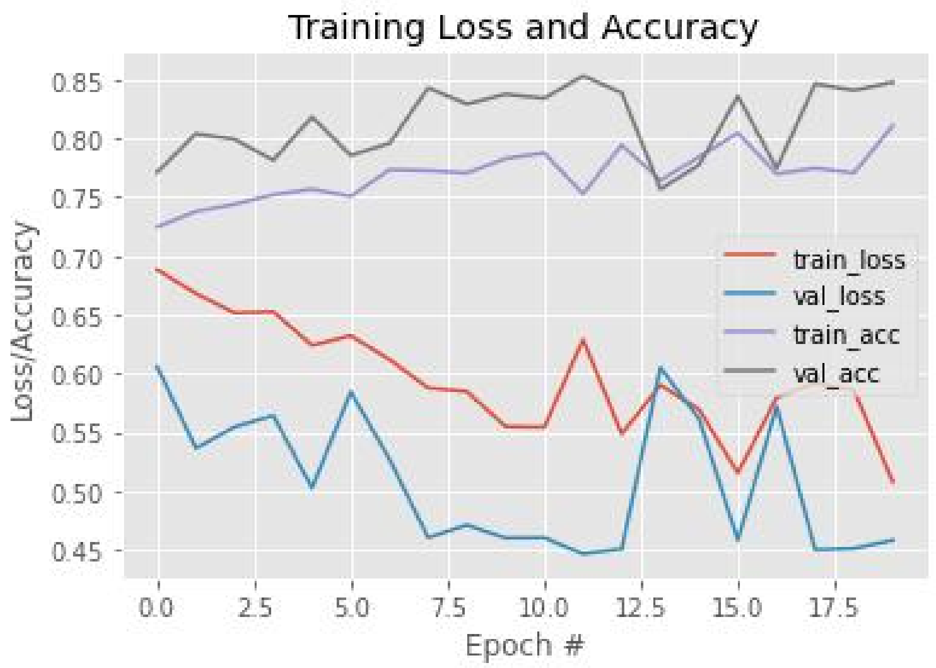


Fig: 7.2.1 Training ,testing accuracy and loss

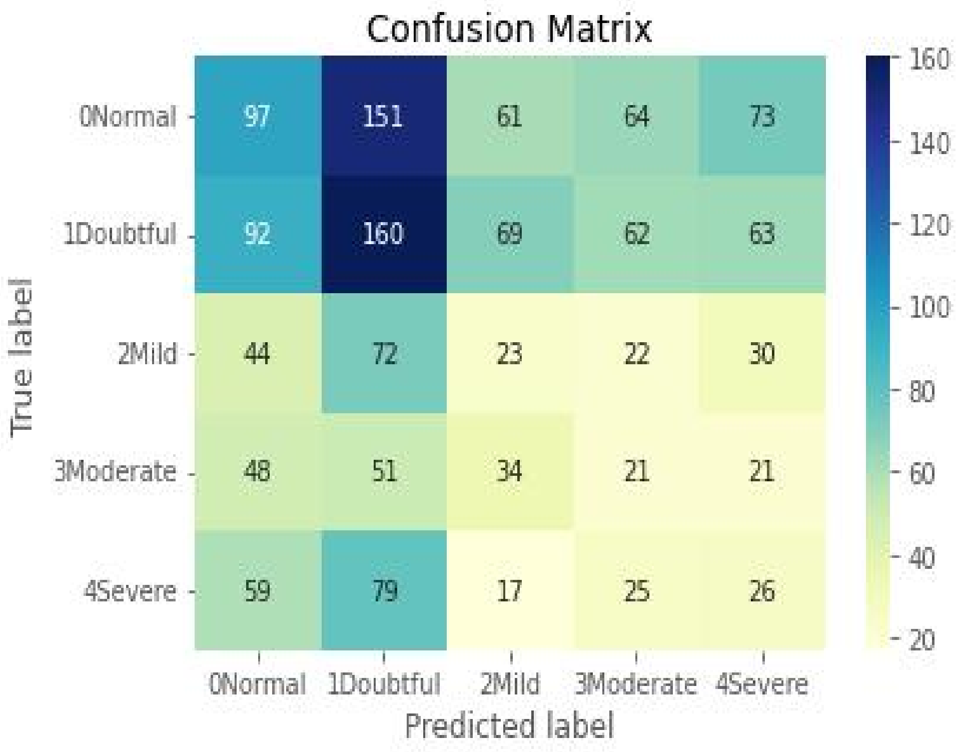


Fig: 7.2.2 Confusion matrix for VGG19 model

7.3 VGG16

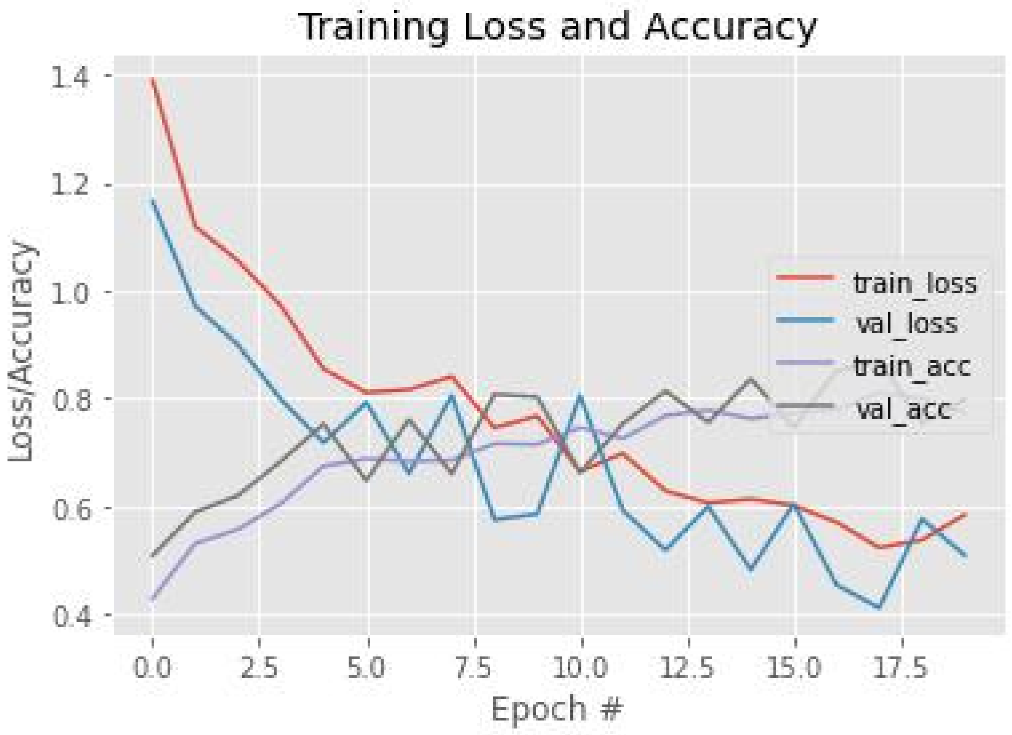


Fig: 7.3.1 Training ,testing accuracy and loss

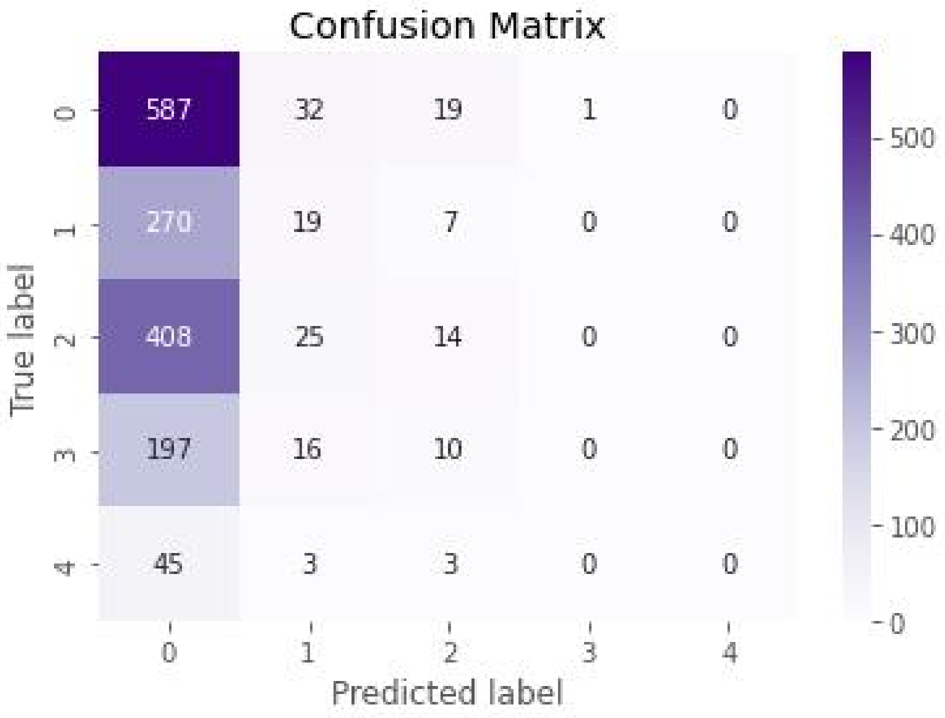


Fig: 7.4.2 Confusion matrix for VGG16 model

7.4 RESNET50

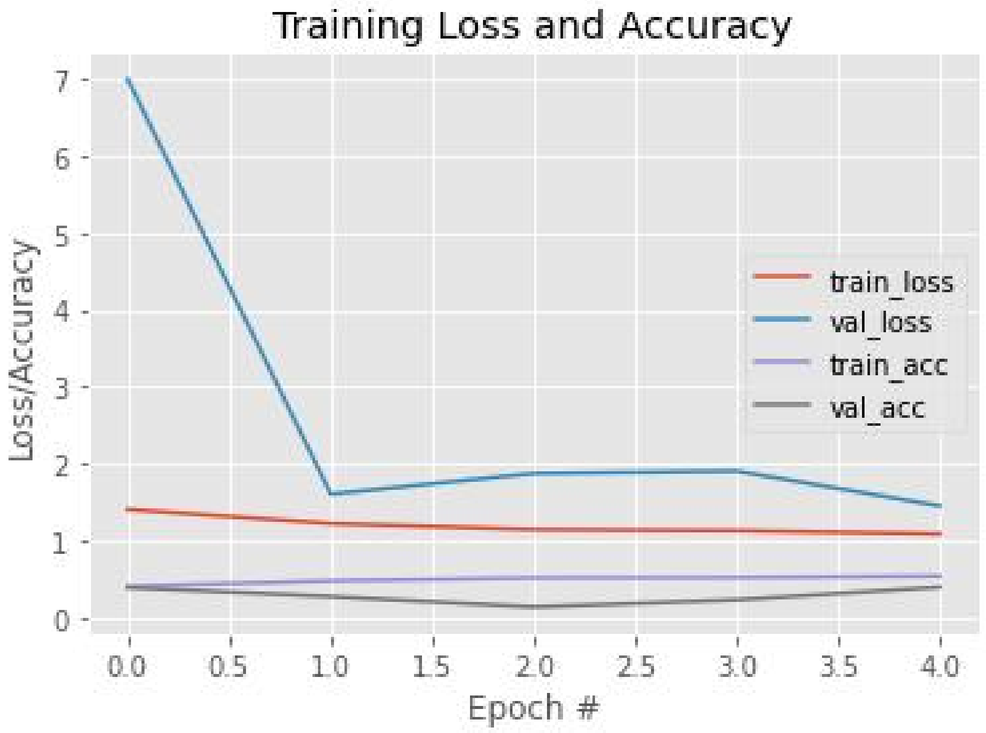


Fig: 7.4.1 Training ,testing accuracy and loss

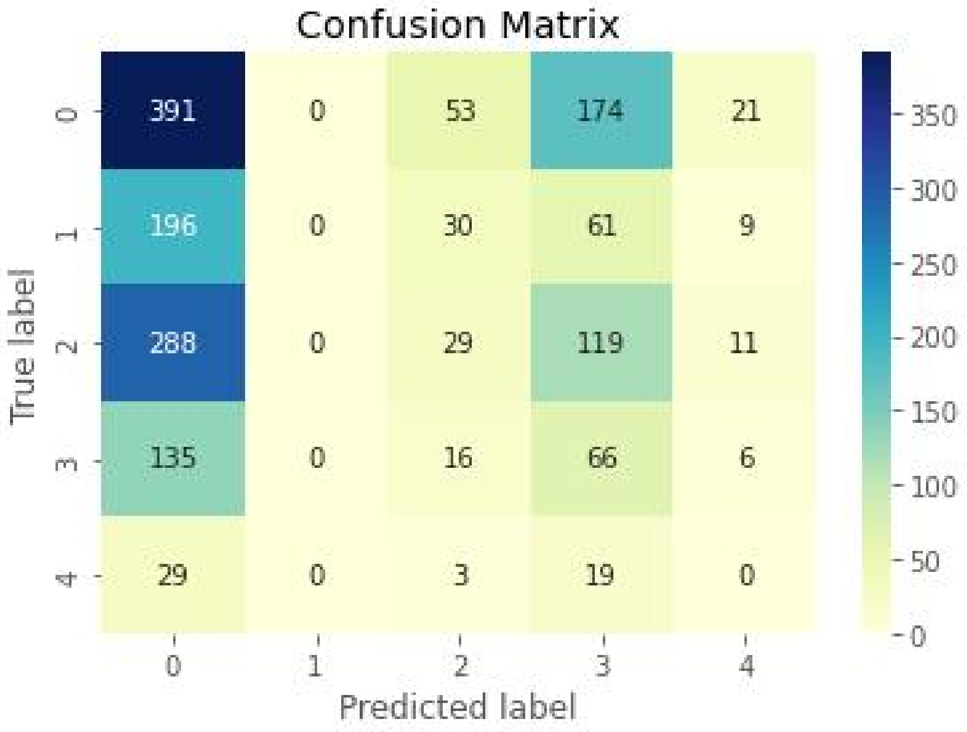


Fig: 7.4.2 Confusion matrix for resnet50 model

7.5 INCEPTION V3

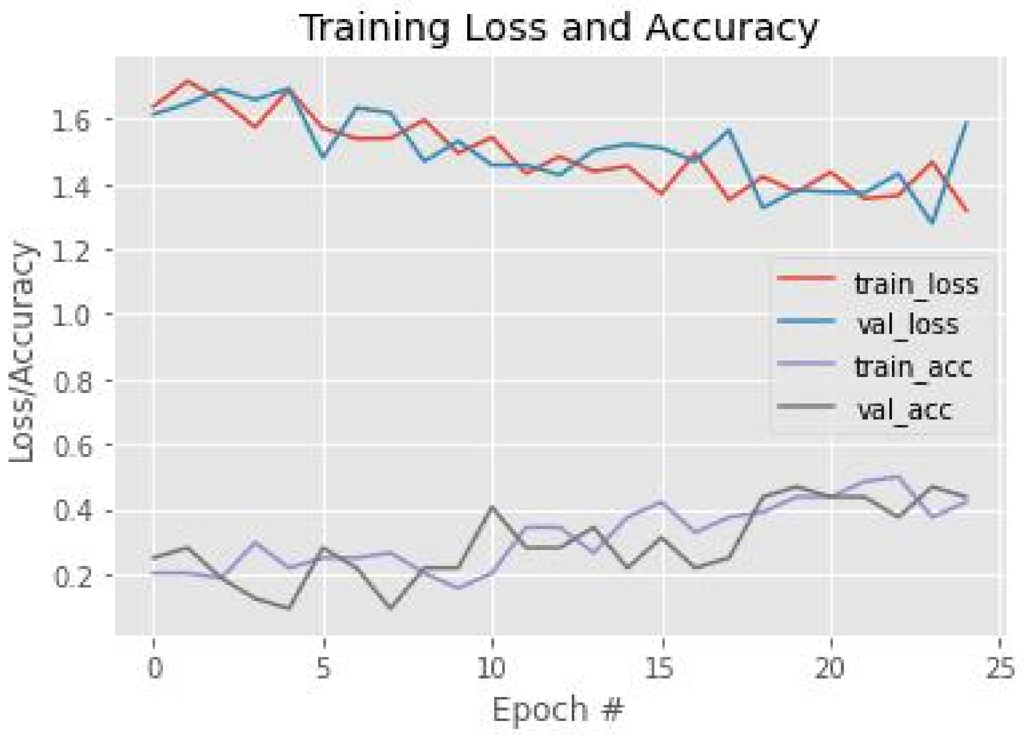


Fig: 7.5.1 Training ,testing accuracy and loss

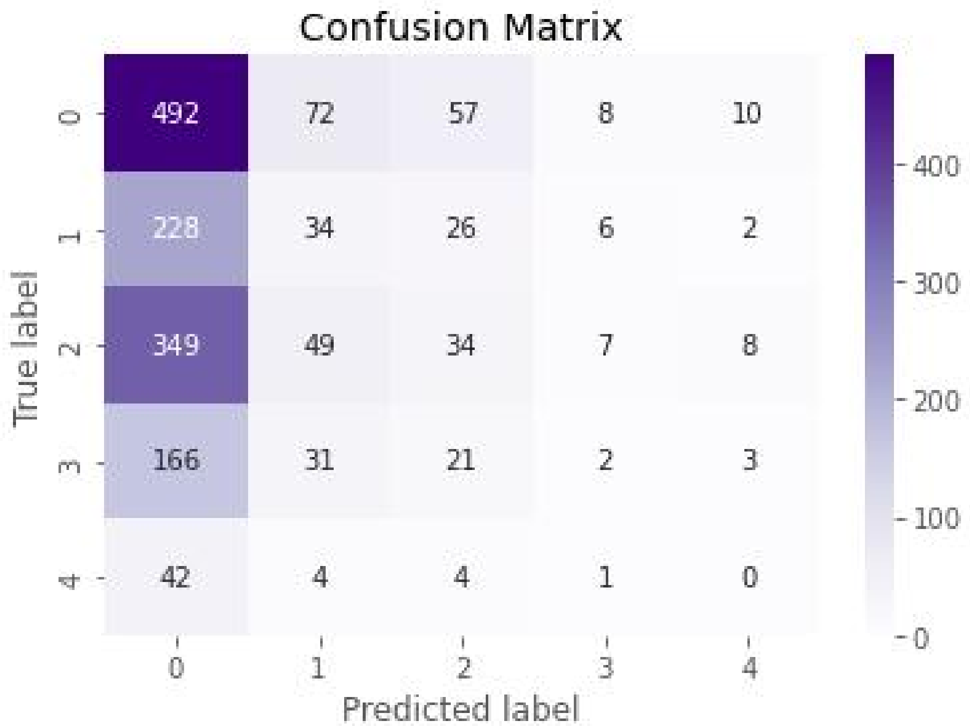
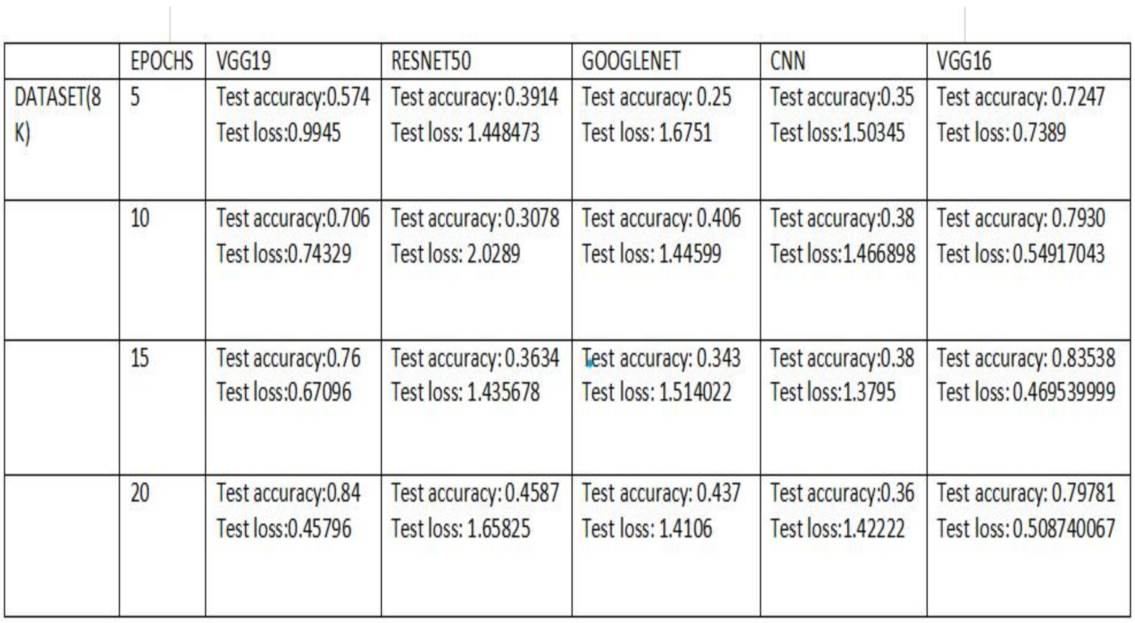


Fig: 7.5.2 Confusion matrix for googlenet model

7.6 Accuracy And Loss For All Models



	EPOCHS	VGG19	RESNET50	GOOGLENET	CNN	VGG16
DATASET(8 K)	5	Test accuracy:0.574 Test loss:0.9945	Test accuracy: 0.3914 Test loss: 1.448473	Test accuracy: 0.25 Test loss: 1.6751	Test accuracy:0.35 Test loss:1.50345	Test accuracy: 0.7247 Test loss: 0.7389
	10	Test accuracy:0.706 Test loss:0.74329	Test accuracy: 0.3078 Test loss: 2.0289	Test accuracy: 0.406 Test loss: 1.44599	Test accuracy:0.38 Test loss:1.466898	Test accuracy: 0.7930 Test loss: 0.54917043
	15	Test accuracy:0.76 Test loss:0.67096	Test accuracy: 0.3634 Test loss: 1.435678	Test accuracy: 0.343 Test loss: 1.514022	Test accuracy:0.38 Test loss:1.3795	Test accuracy: 0.83538 Test loss: 0.469539999
	20	Test accuracy:0.84 Test loss:0.45796	Test accuracy: 0.4587 Test loss: 1.65825	Test accuracy: 0.437 Test loss: 1.4106	Test accuracy:0.36 Test loss:1.42222	Test accuracy: 0.79781 Test loss: 0.508740067

Fig: 7.6 Accuracy and loss for all models

CHAPTER 8
CONCLUSION

8. CONCLUSION

We have consider the Knee X-ray Images for the Knee Osteoarthritis Detection and its Severity using various models like convolutional neural network and transfer learning algorithms of CNN.We have observed the working of each and every model which are been used in our project.In the implementation phase we have started working with CNN algorithm and we got 46% accuracy and later we have worked on various transfer learning methodologies like resnet50,inception v3 ,vgg16,vgg19.On using of resnet50 model we got 39% accuracy comparatively lower than CNN model.Respectively, on usage of the models like Inception V3,VGG16,VGG19 we have secured 43%,83% and 84% accuracies.

We choose VGG19 and VGG16 as our final models because they achieved the highest accuracy score with 84% and 83% on validation sets.Additionally,the models were able to predict the new image into its corresponding severity degree and had good generalization performances.We also consulted domain experts,who confirmed that the VGG models was a suitable choice for this problem. In conclusion, we selected the VGGs as our final models for knee OA detection and its severity.

CHAPTER 9

FUTURE SCOPE

9. FUTURE SCOPE

In future to extension of these we can use more images of dataset for better performance and calculating the accuracy of both training and testing datasets. We can also use different deep learning models for performing the detection and specifying the severity of knee. We can also include more number of convolutional layers and pooling phases where the model can learn more and extract the more features form the image dataset and also usage of more fully connected layers with dropout between 0.5 to 0.8 .

Where different models and usage of larger dataset can bring us to understand more about the outcome. And there may be having of using different epochs for the model which outputs different accuracies and also we can observe the fitness of model for our problem.

CHAPTER 10

BIBILOGRAPHY

10. BIBILOGRAPHY

1. H. Oka, S. Muraki, T. Akune, A. Mabuchi, T. Suzuki, H. Yoshida, S. Yamamoto, K. Nakamura, N. Yoshimura, and H. Kawaguchi, "Fully automatic quantifification of knee osteoarthritis severity on plain radiographs," *Osteoarthritis and Cartilage*, vol. 16, no. 11, pp. 1300– 1306, 2008.

2. Antony, J., McGuinness, K., Connor, N.E., Moran, K.: Quantifying radiographic knee osteoarthritis severity using deep convolutional neural networks. In: Proceedings of the 23rd International Conference on Pattern Recognition. IEEE (2016).

3. Cai, J., Lu, L., Zhang, Z., Xing, F., Yang, L., Yin, Q., 2016. Pancreas segmentation in MRI using graph-based decision fusion on convolutional neural networks. In: International Conference on Medical Image Computing and Computer Assisted Intervention, Springer, pp. 442–450.

4. Yoo TK, Kim DW, Choi SB, Park JS (2016) Simple scoring system and artifificial neural network for knee osteoarthritis risk prediction: a cross-sectional study. PloS ONE 11(2):e0148724.

5. Chen, P.; Gao, L.; Shi, X.; Allen, K.; Yang, L. Fully Automatic Knee Osteoarthritis Severity Grading Using Deep Neural Networks with a Novel Ordinal Loss. *Comput. Med. Imaging Graph.* 2019, *75*, 84–92.

6. Kim, D.H.; Kim, S.C.; Yoon, J.S.; Lee, Y.S. Are There Harmful Effects of Preoperative Mild Lateral or Patellofemoral Degeneration on the Outcomes of Open Wedge High Tibial Osteotomy for Medial Compartmental Osteoarthritis? *Orthop. J. Sport. Med.* 2020, *8*, 2325967120927481.

7. Tiulpin, A.; Saarakkala, S. Automatic Grading of Individual Knee Osteoarthritis Features in Plain Radiographs Using Deep Convolutional Neural Networks. *Diagnostics* 2020, *10*, 932.

8. Postler, A.; Luque Ramos, A.; Goronzy, J.; Günther, K.P.; Lange, T.; Schmitt, J.; Zink, A.; Hoffmann, F. Prevalence and Treatment of Hip and Knee Osteoarthritis in People Aged 60 Years or Older in Germany: An Analysis Based on Health Insurance Claims Data. *Clin. Interv. Aging* 2018, *13*, 2339–2349.

9. Wang, Y.; Wang, X.; Gao, T.; Du, L.; Liu, W. An Automatic Knee Osteoarthritis Diagnosis Method Based on Deep Learning: Data from the Osteoarthritis Initiative. *J. Healthc. Eng.* 2021, *2021*, 5586529.

10. Oka H, Muraki S, Akune T, Mabuchi A, Suzuki T, Yoshida H, Yamamoto S, Nakamura K, Yoshimura N, Kawaguchi H (2008) Fully automatic quantification of knee osteoarthritis severity on plain radiographs. Osteoarthr Cartil 16(11):1300–1306

11. Juefei-Xu F, Naresh Boddeti V , Savvides M (2017) Local binary convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 19–28

12. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, Laak JAVD, Ginneken BV, Sánchez CI (2017) A survey on deep learning in medical image analysis. Med Image Anal 42:60–88

13. Antony J, McGuinness K, O'Connor NE, Moran K (2016) Quantifying radiographic knee osteoarthritis severity using deep convolutional neural networks. In: 2016 23rd international conference on pattern recognition (ICPR). IEEE, pp 1195–1200

14. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

15. Liu, S., Yang, J., Huang, C., Yang, M.H.: Multi-objective convolutional learning for face labeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3451–3459 (2015)

16. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)

Website Reference

- **https://link.springer.com/chapter/10.1007/978-3-319-62416-7_27**
- **https://www.sciencedirect.com/science/article/abs/pii/S0895611118304956**
- **https://link.springer.com/article/10.1007/s11547-022-01476-7**
- **https://www.sciencedirect.com/science/article/abs/pii/S0895611119300035**
- **https://pubs.rsna.org/doi/full/10.1148/ryai.2020190065**