DETECTION OF ARTHRITIS IN HAND USING CONVOLUTIONAL NEURAL NETWORK

A PROJECT REPORT

Submitted by

PRIIYADHARSHINI M	**(18EUEC132)**
SURUTHI R	**(18EUEC165)**
THRISHA R	**(18EUEC170)**

In partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
AFFILIATED TO ANNA UNIVERSITY CHENNAI
ACCREDITED BY NAAC WITH `A` GRADE

MAY 2022

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)

(Approved by AICTE and Affiliated to Anna University, Chennai)

ACCREDITED BY NBA and NAAC WITH "A" GRADE

BONAFIDE CERTIFICATE

Certified that this project report **"DETECTION OF ARTHRITIS IN HAND USING A CONVOLUTIONAL NEURAL NETWORK"** is the bonafide work of "**PRIIYADHARSHINI.M(18EUEC132), SURUTHI.R(18EUEC165), THRISHA.R(18EUEC170)"** who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Dr. S. Sophia M.E., Ph.D.	Dr.V.R.BALAJI, M.E,Ph.D.,
	Professor
HEAD OF THE DEPARTMENT	SUPERVISOR
Department of Electronics and Communication Engineering,	Department of Electronics and Communication Engineering
Sri Krishna College of Engineering and Technology,	Sri Krishna College of Engineering and Technology,
Kuniamuthur,	Kuniamuthur,
Coimbatore -641008.	Coimbatore-641008.

This project report submitted for the Autonomous Project Viva-voce examination held on_____________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to thank the management for supporting us during our entire process and encouraging us to be innovative and backing us in everything we did and we would like to thank our beloved principal **Dr. J. Janet**, for allowing us to do this internship and providing required time to complete the same.

We express our sincere thanks to our beloved Head of the Department **Dr. S. Sophia,** for her encouragement in our endeavor.

We are thankful to our project coordinator **Dr.V.R.Balaji**, Professor and **Ms.D.V.Soundari**, Assistant Professors, Department of Electronics and communication Engineering, for their valuable suggestions and guidelines to implement this project.

We deem it our duty, our sincere admiration and heartfelt gratitude to our project guide**, Ms.N.Kalaivani**, professor, Department of ECE, for having effectively guided and supervised throughout this project by imparting his erudite knowledge and personalized guidance blended with exemplary patience and encouragement.

ABSTRACT

Osteoarthritis is one of the prime causes of infirmity in elderly and overweight people. Osteoarthritis is a joint disease that mostly affects the cartilage. Cartilage helps the easy glide of bones and obstructs them from rubbing each other. In Osteoarthritis cartilage is ruptured due to which bones rub each other causing severe pain. The current strategy for the evaluation of Osteoarthritis includes clinical investigation and medical imaging techniques. In this project, we want to detect and classify Osteoarthritis in Hand from medical images using Deep Features to categorize the images. In this project, the problem of noises can impact on the detection and classification of target area in images and because of it the irrelevant features could be selected from the medical images. This project will also focus on handling the huge amount of image data by utilizing some High-Performance Computing. This Project also discovers the Magnetic Resonance imaging (MRI) techniques for detection and classification of Osteoarthritis in descriptive and comparative manner. Thus, an integrated discussion of various detection techniques, feature extraction techniques and classification schemes regarding Osteoarthritis is done in a scientific way.

Keywords: Medical Imaging, MRI, Osteoarthritis, OA detection, classification, Deep Learning

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATION

OA	Osteoarthritis
CT	Computed Tomography
SSD	Single-Shot Multibox Detector
AC	Articular Cartilage
RA	Rheumatoid Arthritis
IDE	Integrated Development Environment
CNN	Convolutional neural network
AI	Artificial intelligence
GPU	Graphics Processing Unit
MAP	Mean Average Precision
MRI	Magnetic Resonance Imaging

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Recent advances in artificial intelligence have led to fully automated work-flows that often exceed human performance. State of the art neural networks can detect the objects in the images and classify them into thousands of categories more accurately and magnitudes faster than humans. They translate texts from multiple languages, drive cars autonomously through cities, and detect malware in computer systems. In most of these cases, they have been trained on tens of thousands, or even millions, of data samples. Neural networks have also found great success in the field of medical image analysis where data sets are often much smaller. Although the same techniques can be applied, one is often confronted with a different set of challenges.

Medical imaging is the process of creating visual representations of the internal structures hidden by the skin and bones. It is the technique where we can reveal the interior of the body for clinical diagnosis and medical intervention. It is the part of biological imaging and incorporates radiology which uses the imaging technologies of X-ray, Magnetic Resonance Imaging (MRI), Ultrasound, Computed Tomography (CT) etc. Osteoarthritis is one of the most common form of arthritis disease that is seen mostly in females, overweight and elderly people. Osteoarthritis (OA) isa joint disease that mostly affects cartilage. Cartilage is the protective connective tissue that covers the end of bones in a joint. Healthy cartilage allows easy glideof bone in the joint and prevents them from rubbing each other. In Osteoarthritis the top layer of cartilage breaks down and wears away. This allows the bones to rub each other causing pain. It commonly affects the joint in the Hand, hip, spine and feet. The most common cause of osteoarthritis of the Hand is age. There are two types of OA, Primary OA seen in aged people due to genetic reasons or aging. Secondary OA tends to show up earlier in life due to some injury, diabetes, obesity, athletics or patients with rheumatoid arthritis. The sample of normal and affected Osteoarthritis Hand image is shown

The main symptoms of OA are pain and difficulty in joint motion, reduced function and participation restriction, Joint stiffness in the morning or after rest. Currently evaluation of OA is based on clinical examination, symptoms and simple radiographic assessment techniques (X-ray), MRI, CT etc. While several other methods have been proposed, Kellgren-Lawrence (KL) system is a validated method of classifying individual joints into 5 grades. Table 1.1 below shows the different grades of OA disease.

KL Grades	OA Analysis
Grade 0	No Radio-graphic features of OA present
Grade 1	Doubtful OA (narrowing of joint space)
Grade 2	Mild OA (definite narrowing of joint space)
Grade 3	Moderate OA (multiple osteophytes, sclerosis)
Grade 4	Severe OA (large osteophytes, sever sclerosis, bone deformity)

The common X-ray findings of OA include destruction of joint cartilage; joint space is diminished between adjoining bones and bone spur formation. MRI scans may be ordered when x- rays do not give clear reason for joint pain or when the x- ray suggests that other type of joint tissues can be damaged. The current methods used for clinical diagnosis of Osteoarthritis are not accurate enough to efficiently measure the quality and evolution of Osteoarthritis. Thus we require more significant methods and algorithms which are multifactoral to access the parameters and progression of Osteoarthritis.

Detection and Classification of Osteoarthritis in Hand from medical images is one of the active fields in computer vision. Image classification is one of the most commonly studied problems in computer vision. The goal of this field is detecting all the objects a given image. Since the Convolution Neural Networks (CNN) can detect the objects with more than 90% of accuracy, we can use a fine-tuned neural network to detect an object. In the respective for above-mentioned issues, this work will implement Deep Features techniques to detect and classify the Osteoarthritis from medical images and also incorporate a large database for better accuracy. This project focuses on the various machine learning techniques for the classification of OA disease from the medical images.

Musculoskeletal diseases and articular disorders are one of the major health problems in recent years and affect especially the aging population. The human Hand joint is commonly affected by osteoarthritis (OA), a degenerative disease that is the primary cause of chronic disability. It is a highly prevalent joint disease, causing pain and disability in older people, and it can be characterized by progressive degradation of the diarthrodial joint tissue. OA can affect the joints of the spine, fingers, thumbs, hips, Hands, and toes as depicted in Figure 1.1, but it is most prevalent in Hand andhip joints. Detection and progress monitoring of Hand OA can be done by measuring biochemical and anatomical changes associated with the tissues such as articular cartilage, ligaments, meniscus, synovial fluid, and subchondral bones.

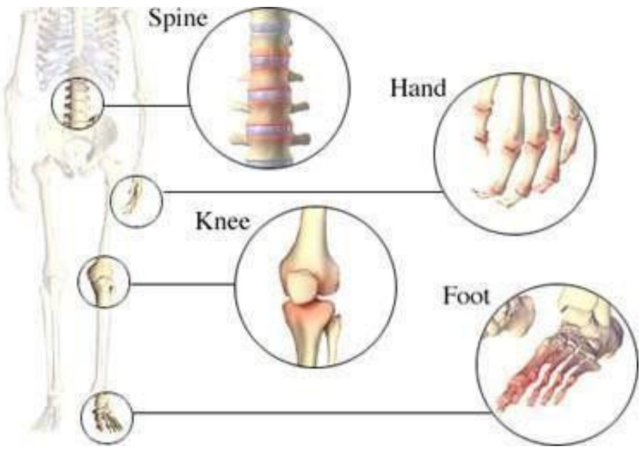


FIGURE 1.1: Locations of various human body joints affected by osteoarthritis

An early detection and monitoring of osteoarthritis is possible by measuring pre-structural and structural changes associated with the tissues such as articular cartilage, meniscus, synovial fluid, and subchondral bones. Soft tissues within the di- arthrodial joint have been successfully used to detect and monitor Hand OA, changes associated with the subchondral bones are a promising imaging biomarker for as- sessing Hand OA conditions.

The worldwide prevalence estimate for symptomatic OA is 9.6 % among men and 18 % among women. In addition, progression of early OA can take place within 10 years of a major injury. In case of injury at age 15, young people may have OA as early as age 25.

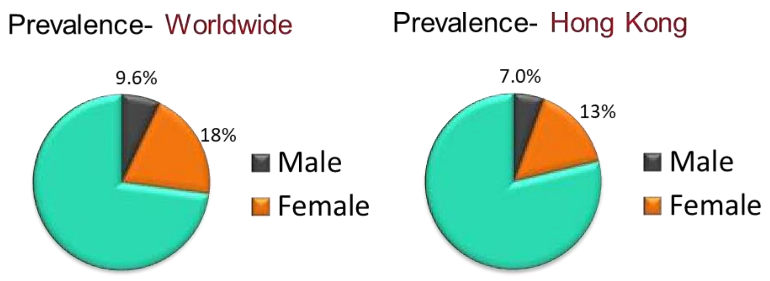


FIGURE 1.2: Prevalence of Osteoarthritis Worldwide and Hong Kong

According to the studies by World Health Organization for the year 2000, Hand OA is most prevalent in developing countries in Europe (EU BC) (13.3%), Eastern Mediterranean Region (30% aged above 60) and South-East Asia Region (SEA) (7.9%) for the age group between 30 – 80 years. In Hong Kong, prevalence of osteoarthritis is showing similar trend as other part of world. A study in 2000 reported that among Hong Kong people aged 50 and above, 7 percent of men and 13 percentof women suffered from osteoarthritis.

Problem Definition

Object detection is a general term for computer vision techniques for locating and labeling objects. Object detection techniques can be applied both to static images or moving images. Computer vision techniques are already in wide use in medical areas today, as they can offer valuable insight about various diseases to detect efficiently. In this project we will be solving an object detection and image classification problem, where the goal will be to detect the OA affected area in the OA Hand MRI images. Detection is the process of finding out the particular objects in the images, in this case finding out the OA affected area from Hand MRI images. Whereas, classificationis a process of classifying whether the output image is OA affected or Normal.

Object detection is widely used in the world of medical to detect the diseases. Neural network based classifiers are used together with other object detection techniques. During this project we have explored the modern open source based solutions for object detection in medicals, in this case for detecting OA disease. Tensor Flow Object Detection API, an open source framework for object detection related tasks, was used for training and testing an SSD (Single-Shot Multibox Detector)

with Mobilenet- model. The model was tested as pre-trained and with fine-tuningwith a dataset consisting of OA MRI images. The sample detected OA region in Hand MRI is shown.

Open source library such as TensorFlow Object Detection API offers easy-to-use, open source frameworks where pre-trained models for object detection (such as ones downloadable from the TensorFlow model zoo) reach high accuracy in detecting various object from humans to tv monitors. However, the dataset of MRI images used in this project made it important to train the models to work with the specific data. For this project, TensorFlow Object Detection API was chosen due to its popularity in object detection and for its easy-to-approach nature. In this project, TensorFlow Object Detection API was tested for detection of OA disease. The model used within the API is a SSD model with Mobilenet. The pretrained model is trained and tested with our own data which consisted of MRI images of Hand OA.

Osteoarthritis

The development of OA in the Hand is due to a heterogeneous group of conditions involving all the components in diarthrodial joint but mainly associated with defective integrity of articular cartilage (AC), meniscus damages and related changes in the underlying bone at the joint margins. An early detection and monitoring of osteoarthritis is possible by measuring pre-structural and structural changes associated with the tissues such as articular cartilage, meniscus, synovial fluid, and subchondral bonres In the assessment of osteoarthritis, pre-structural/biochemical composi- tions (water content, proteoglycan content and collagen) are mainly measured in the articular cartilage tissue to diagnose the disease at early stages followed by the mea- surement of anatomical (thickness, volume, surface area) changes that occurs either at early or during the progression stages. Similarly, quantitative MR of meniscus has been used to measure biochemical (collagen-PG) changes during the early onset ofosteoarthritis followed by its morphological (flap, or complex tears; meniscal macer- ation; or destruction) changes to monitor progression. Although soft tissues within the diarthrodial joint have been successfully used to detect and monitor Hand OA, changes associated with the subchondral bones are a promising imaging biomarker for assessing Hand OA condition. Researchers have demonstrated that the changes in subchondral bone shape morphometry, osteophytes, surface area, and bone marrow lesions are important to assess OA progression, and they can effectively be measured using magnetic resonance (MR) imaging.

Quantitative MRI can be used to diagnose, assess, and monitor diseases such as osteoarthritis by identifying morphologic changes in the Hand and calculating quantita- tive values such as T1 rho, T1/T2 relaxation times. However, the clinical application of quantitative MRI has been hindered by the requirement for time-consuming post-processing of images, particularly for segmentation of the joint and musculoskeletal tissue. Generally, measurement of physiological changes in AC using quantitative MR techniques requires segmentation of cartilage region followed by generating a technique specified map which is being overlaid to measure specific values in Hand tissue. In most cases, the segmentation of AC (either the whole or different compart-ments) is performed either by an expert or using semi-automated methods which are tedious and time-consuming that may also lead to reliability issues. If it’s done manually, it can take a skilled technician around three hours to do an image, depending on the quality of the image and what actually is being segmented. Thus, segmentation of Hand joint tissues such as cartilage, bone and meniscus from medical images is an important perspective to researchers and clinicians for the development of biomarkers, Hand implant procedures, modeling of the Hand to predict kinematics and understanding the physiological phenomenon in healthy Hand joint tissues. As a result, the team sought to utilize advanced computation approaches including image/pattern analysis techniques in combination with deep learning to develop a fully automatic segmentation technique for bone, cartilage and meniscus. The developed technique would accurately extract quantitative measurements in terms of cartilage volume and thickness, bone area/shape & attrition, meniscus damages as well as biochemical compositions by estimating Spin-Lock relaxation times on Hand MRI.

CHAPTER 2
LITERATURE REVIEW

Automatic Prediction of Rheumatoid Arthritis Using CNN

Rheumatoid arthritis can be defined as a chronic inflammatory disorder affecting the joints by damaging the tissue of the body. Therefore, an effective system analysis is necessary for the identification and detection of rheumatoid arthritis by hand, especially during its development or pre-diagnostic stages.This project is designed to develop an intelligent system to detect rheumatoid arthritis of the hand using image processing techniques and a neural network of convolution.The system comprises of two main phases. The image processing phase is the first stage in which images are processed using image processing. These techniques include preprocessing, image segmentation and feature extraction using gabor filter. The second phase the extracted features being used as inputs for the neural convolution network, which classifies the hand images as normal or abnormal (arthritic). Classification is carried out based on the CNN algorithm, which involves the training of the network with normal and abnormal hand images. The system was tested on the same number of images as the testing set and the experimental results showed that a recognition rate of 83.5% respectively.

This project is designed to develop an intelligent system to detect rheumatoid arthritis of the hand using image processing techniques and a neural network of convolution. The system comprises of two main phases. The image processing phase is the first stage in which images are processed using image processing. These techniques include pre-processing, image segmentation and feature extraction using gabor filter. The system was tested on the same number of images as the testing set and the experimental results showed that a recognition rate of 83.5% respectively.

Arthritis Identification from Multiple Regions by X-Ray Image Processing

The x-ray image processing is an effective technique to distinguish between the major types of arthritis: Osteoarthritis Arthritis (OA) and rheumatoid Arthritis (RA). The degenerative bone disorders are diagnosed using X-ray detection. X-ray scans alone are insufficient to detect the type of arthritis. Image processing can aid in improving the diagnosis. The classification is done on the basis of differentiation in the region properties and boundary areas that can be identified using MATLAB. These properties were used to better understanding the variation in the knee, hands and neck region. The study holds potential as a diagnostic tool for arthritis identification through x-rays. It could pave way to differential diagnosis in future

The only means of detection of arthritis is by X-Ray studies in Pakistan thus there is room for research on better or improved methods of assessment and prediction of the disease type and degree. There is often a mismatch between the stage of OA depicted and the actual degree of pain and disability observed by the patient while the X-Ray cannot solely depict clear distinction amongst different forms of arthritis. MRI (magnetic resonance imaging) on the other hand provides high-resolution internal body tissues images. A strong magnet passes a force through the body to create an image. MRI tests are used if x-ray analysis is insufficient and other joint tissues such as a ligament are damaged. NIAMS-researchers are working on a technique called micro computed tomography (micro CT). It is a high-resolution, three-dimensional x-ray image creating technique which can identify the content and distribution of proteoglycans. This is involved in studying the thickness and composition of cartilage, thus helps to monitor the progress and treatment of OA. The technique is limited to animal studies and success rate in humans is still a far cry. The diagnosis by conventional X-Ray is a less expensive technique as compared to MRI and micro CT, thus by designing a detection algorithm on X-Ray images in MATLAB the identification can be made more accurate.

Arthritis is a bone related disabling disorder. The field is very diverse and it is hard to classify the kind of arthritis by means of imaging alone. The image processing techniques are thus used to better contemplate the differences and device a possible diagnostic tool. The Osteoarthritis (OA) is degenerative disorder and Rheumatoid Arthritis (RA) is mainly genetic disorder but the damage to the bones is of fine difference between the two conditions. The region properties such as centroids, orientation, major axis length, minor axis length, extrema and eccentricity are used in the above study to statistically analyze the differences in the knee, hands and neck regions of the normal, OA and RA patients. The centroid, orientation, area and boundary identification on the images using MATLAB helps with better insight. The symmetrical and asymmetrical wear and tear is one noteworthy point. The orientation is more evident in the case of RA than OA. The centroids are more likely to form in the greater degree of bone spurring. These findings can help understand better areas of concern to help in formulation of a precise classifier. The image processing technique can be used to better understand the variation amongst the different type's thus enabling stronger grip on the possible pre-requisites for an Arthritis classifier tool. The SVM train-test can be used to program the classifier to recognize the similarities amongst the same kind of arthritis and differentiate from the other kind. This would serve as a differential diagnostic tool to aid the identification of specific type of arthritis from the x-ray image processing.

Rheumatoid arthritis detection using image processing

Rheumatoid Arthritis (RA) may be a general disease characterized by inflammation, discomfort, and tenderness of the joints and might involve additional body part organs in severe cases. Leading to increased vascular disorder in the zone of inflammatory tissue, joint autoimmune lesions are associated with elevated fever. The detection of RA usually involves blood sample tests. This thesis proposes a novel methodology of detection by processing the Xray images. This automated system requires clear Xray images, which after preprocessing and segmentation using Support Vector Machine implemented via MATLAB gives a clear classification about the abnormal and normal images. Different output parameters were used to assess separation tasks. The accuracy of the model section has improved to the use of an optimized SVM network. The proposed model was effective in accurately separating the samples.

The calculation effect can be observed to aid in the diagnosis and analysis of arthritis. Depending on the size of the cartilage, this procedure is easy and efficient in achieving the target of early bone detection. The Gray Level Co-occurrence Matrix was used to retrieve the features. The whole element was taken into consideration for separation, and it was given to the separator for that reason. Support vector machine division was used to get the total result.

Sungroh Yoon and Taehoon Lee have introduced a sequencing software to delete unwritten areas from pre-messenger ribonucleic acid (RNA) copies. Finding target sequences is a crucial machine learning activity that aids in the identification of basic genes as well as the understanding of how various proteins are made. Existing prediction repetition approaches have shown good results, but they are also moderately stable and inaccurate. A comprehensive theory of a mechanism based on a predictive network of computational splice junction is presented in this article. The idea involves a novel method of training Boltzmann's equipment collection for horizontal inequality prediction. The proposed method overcomes the limitations of distinct variants and allows for the construction of datasets with distinct features. Using individual social data sets, this approach was shown to have greatly increased accuracy and decreased running time as compared to other methods. The suggested approach is more efficient at handling false interaction signals and is less vulnerable to the duration of the input chain.

P. Thangam, K. Thanushkodi, and Thushika Mahendiran have created a curriculum that includes endocrinological disorders in children that are seen in many countries worldwide and vary in diameter and severity in various age ranges and races. Changes in diet and eating habits also correlate to endocrine disorders, necessitating the development of a device that can predict those issues ahead of time. In the treatment and diagnosis of endocrine diseases, osteoporosis is a popular technique. It may also be seen as a sign of a medicinal effect. In pediatric medicine, it's important for identifying growth hormone or even genetic abnormalities. The left-hand connector was used to determine bone age, which was then applied to time. The disparity between the two demonstrates the irregularity.

As per the preceding survey, researchers have investigated various methods on the detection and classification of Osteoarthritis using different Hand images like X- ray, MRI etc. Image segmentation method has a great importance in most medical imaging applications. In medical imaging the segmentation methods are classified into two basic groups: pixel based (includes thresholding, region growing region merging etc) and geometry based (includes deformable models, active contours ac- tive appearance models). Therefore to know the proper progression & extremity ofthe disease medical imaging is carried out in terms of X-ray imaging or Magnetic Resonance imaging. As per the survey,

Lior Shamir et al., have used joint detection algorithm and feature extraction technique to analyze the Hand X-ray image. The algorithm finds the joint and separates it from rest of the images. The features were computed by Zernike features, Multi-scale histograms, first four moments, Tamura texture features etc. The features extracted were classified using weighted Nearest Neighbor rule. The authors concluded that 95% of moderate OA was differentiated from normal and 80% of minimal OA was differentiated from normal.

M.Subramonium et al., have used edge detection algorithm, to detect edges of Hand X-ray images in osteoarthritis. The features computed were femur, tibia and
patella cartilage. The authors have concluded that the proposed algorithm gives sufficiently good results and is very effective in noisy and blur images.

Samir K.Bandyopadhyay et al., have used Local Binary Pattern (LBP) based classification system for the detection of OA using Hand X-ray images. The classification is achieved by computing the histograms of LBP of Hand X-ray images using K- Nearest Neighbor classifier. The authors have concluded with 95.24% of accuracy in detecting normal or abnormal Hand image and 97.37% of accuracy in detecting medium or worst cases.

Prafull Sharma et al., have used different Image segmentation algorithms on X-ray bone images. The discrete step, Watershed segmentation and Otsu’s segmentation are used to analyze the abnormalities and problems associated with bone structures. The authors have concluded that discrete step algorithm provides quick and efficient results.

Bindushree R et al., have used different image processing techniques to measure the joint space width in Hand x-ray images. Different techniques used are contrast enhancement, histogram equalization, canny edge detection algorithm and thresholding for extraction & computation of features. The authors have concluded that the Joint Space Width of the Hand x-ray image is compared with the standard Width (4.8 for women and 5.7 for men) and that image is said to be normal case or os- teoarthritis case.

Subromoniam M et al., have used computer aided diagnosis for the detectionof OA using X-ray images. Haralick feature extraction technique for computation & SVM classifier with the kernel functions are used for the detection & classification of OA. They have concluded that the algorithm had a sufficient good result with accuracy of 99% in the diagnosis of bone disorders caused by OA.Dian Pratiwi ,have used artificial neural back propagation method for measuring the severity of Osteoarthritis disease. In their work the whole processing is di- vided into three steps, image processing, feature extraction and classification usingartificial neural network process. They have concluded with 66.6% of classification rate.

Jessie Thomson et al, features, simple pixel features etc. The experiment showed that combining shape & texture based classifiers resulted in 84.9% of accuracy as compared to shape alone.Tati L.Mengko et al., have used machine vision system for osteoarthritis assess- ment. In their proposed method image segmentation uses edge detection method to determine the region of joint space. The feature computed is the distance between femur & tibia bone. The classification of normal & affected OA is obtained from
Radio graphic image using neural network that is later examined with the predefined diagnosis managed by the physician. They concluded with 50% sensitivity, 100% specificity & positive predictive value & 91.84% negative predictive value.
From the survey it is observed that a good number of researchers have worked on Hand X-ray imaging for detection and classification of OA using different approaches and forecasted some promising results with their own datasets. But still there is no such research work done on Hand MRI (Magnetic Resonance Imaging). There is also a scope for using such robust algorithm using different parameters on Hand MRI as MRI, also known as nuclear magnetic resonance imaging, is a scanning technique that uses strong magnetic fields and radio waves to generate a detailed image of thebody's soft tissue and bones. Different robust features are required to be extractedfrom Hand MRI also for the detection of Osteoarthritis. The comparative analysis with promising results were made on Hand x-ray by researchers as predicted and discussed in the related work. But still these algorithms & methods are to be applied using Hand MRI.

For the classification and detection of Hand Osteoarthritis, the segmentation of Hand cartilage and the surrounding bones is a problem which has gained considerable importance in recent years. Segmentation of Hand joint tissues such as cartilage, bone and meniscus from medical images is an important perspective to researchers and clinicians for the development of biomarkers, Hand implant procedures, modeling of the Hand to predict kinematics and understanding the physiological phenomenon in healthy Hand joint tissues. Segmentations of bones are required for computer- based surgical planning of Hand implants. Other applications include modeling of the Hand by finite elements to predict joint kinematics or the understanding of natural variation and physiological effects for healthy joints.

Neural networks are known to be feature selectors, meaning that they will learn to extract information that is relevant to the task. This assumes that the size of the data set and the complexity of the problem enable the network to find correlating features. Based on a series of tests, it was not possible to create a stable algorithm that would predict the OA of an individual based on their Hand MRI and hence the classification between normal Hand MRI and OA Hand would also be predicted.

Summary

In this chapter we have described the survey of various researcher’s works, and studied the related research problems. The background study related to this project and an overview of prerequisites that led to the main task of this project is described in chapter three. It features a section for the necessary medical knowledge as well as a hierarchical derivation to the appropriate machine learning methods. However,
the methodology that we have used in doing this project is described in chapter four.

CHAPTER 3
SYSTEM DESIGN

3.1 BLOCK DIAGRAM:

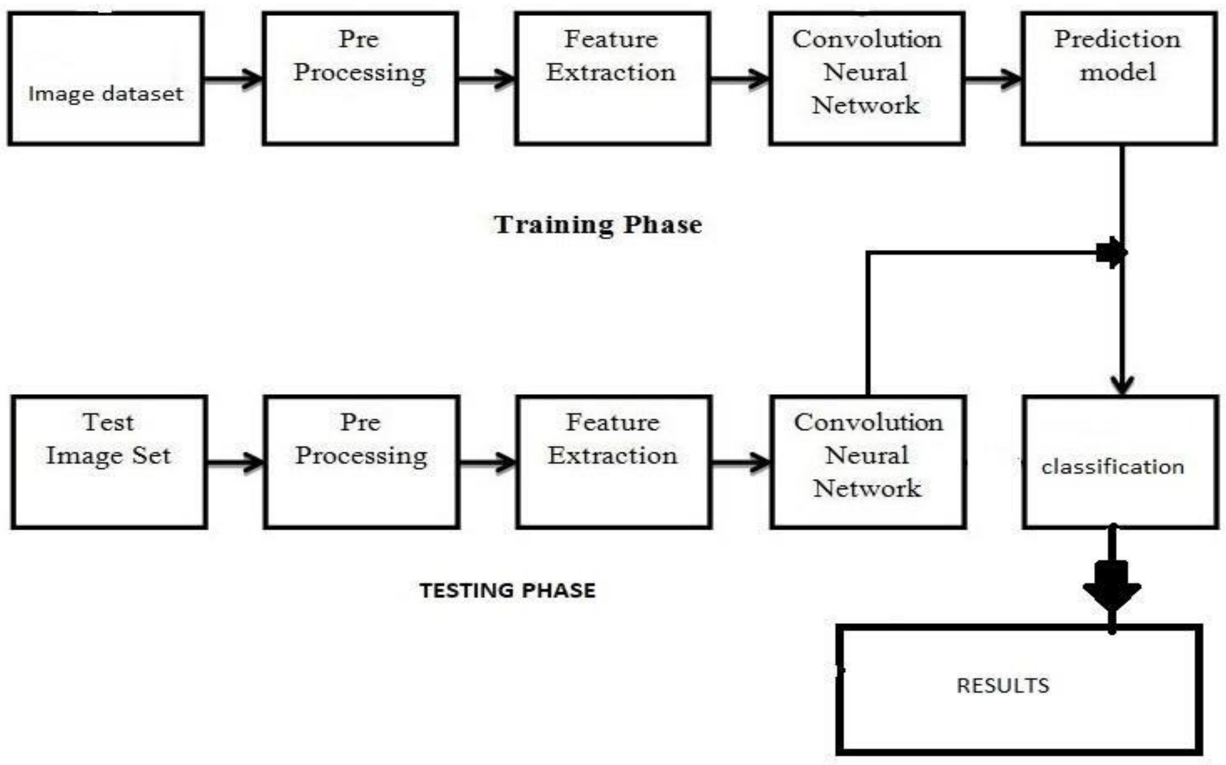


FIGURE 3.1: Overall Block Diagram

For the development of this work, we selected several CNN architectures and trained them end-to-end with hand radiograph images using only their pixel information. With these experiments we seek to evaluate the capacity of the models to learn RA visual features in a similar way a rheumatologist uses in a diagnosis.
Training a model with small amounts of data is a challenging task. To face this problem the machine learning community has been using different procedures to artificially extend datasets.

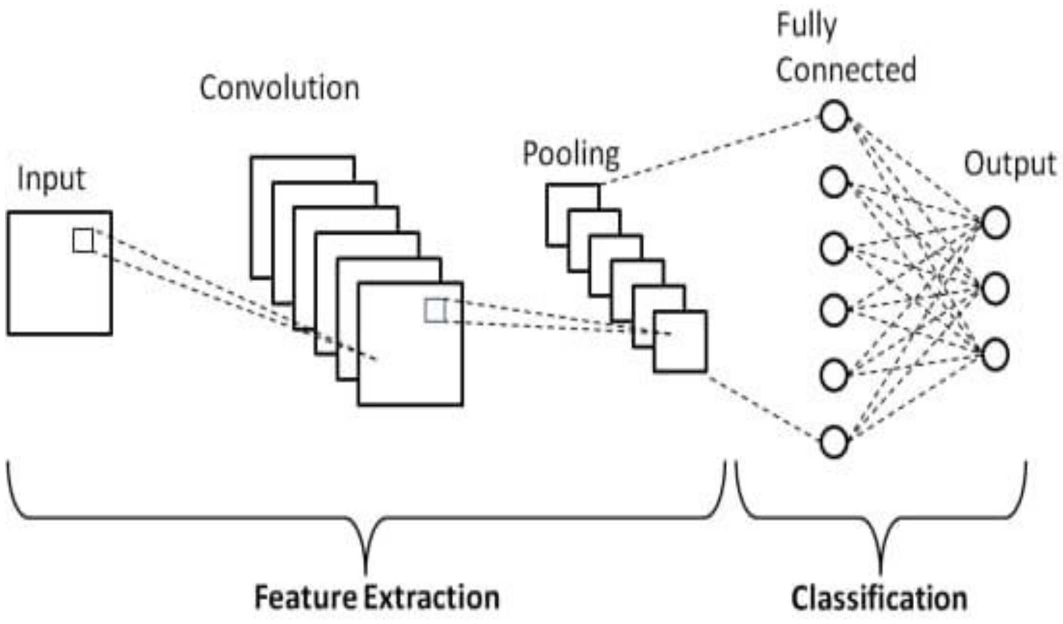


FIGURE 3.2: C o n v o l u t i on a l n eu r a l n et w ork (CN N)

Recognizing the specific phase of RA is a difficult assignment, as human abilities regularly curb the techniques for it. Convolutional neural network (CNN) is the center for hand recognition for recognizing complex examples.

The human cerebrum capacities work in a high-level way, so CNN has been planned depending on organic neural-related organizations in humans for imitating its unpredictable capacities.

This article accordingly presents the convolutional neural network (CNN) which has the ability to naturally gain proficiency with the qualities and anticipate the class of hand radiographs from an expansive informational collection.

The reproduction of the CNN halfway layers, which depict the elements of the organization, is likewise appeared

The result indicates that hand X-rays are rated with an accuracy of 94.46% by the proposed methodology. Our experiments show that the network sensitivity is observed to be 0.95 and the specificity is observed to be 0.82.

Osteoarthritis is the most common type of arthritis. Hand osteoarthritis leads to specific structural changes in the joints, such as asymmetric joint space narrowing and osteophytes (bone spurs).We aimed to develop a computerized method that is capable of determining the structural changes seen in radiography of the hand and to assist practitioners in interpreting radiographic changes and diagnosing the disease.

CHAPTER 4
METHODOLOGY

4.1 PREPROCESSING:

Images come in different shapes and sizes. They also come through different sources. Taking all these variations into consideration, we need to perform some pre-processing on any image data. RGB is the most popular encoding format, and most “natural images” . Also, among the first step of data pre-processing is to make the images of the same size.
Here we have used auto resizing for training to make all the images in the dataset to convert in to same resolution.

4.2 FEATURE EXTRACTION:

The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine’s efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

4.3 CNN:

In deep learning, a convolutional neural network (CNN) is a type of deep neural networks, which deals with the set of data to extract information about that data. Like images, sounds or videos etc. can be used in the CNN for the data extraction. There are mainly three things in CNN. First one is local receptive field and then shared weight and biases and the last one is activation and pooling. In CNN, first the neural networks are trained using a heavy set of data so that the CNN can extract the feature of given input. When the input is given, first image preprocessing is done then the feature extraction occurs on the basis of set of data stored and then the classification of data is done and output is shown as the result.
The CNN can deal with those inputs only for what the neural network is trained and the data is saved.
They are used in image and video recognition, recommender systems, image classification,

medical image analysis, and natural language processing

4.4 DATASET:

One major advantage of using CNNs over NNs is that you do not need to flatten the input images to 1D as they are capable of working with image data in 2D. This helps in retaining the "spatial" properties of images. So here we are using x-ray data base which consist of three categories Frontal, Maxillary and Normal.

4.5 PRE-PROCESSING STEPS

The pre-processing steps were conducted with resizing, patch, and augmentation steps. The first pre-processing step normalizes the size of the input images. Almost all the radiographs were rectangles of different heights and too large (median value of matrix size ≥1,800). Accordingly, we resized all images to a standardized 224×224 pixel square, through a combination of preserving their aspect ratios and using zero-padding. The investigation of deep learning efficiency depends on the input data; therefore, in the second processing step, input images were pre-processed by using a patch (a cropped part of each image). A patch was extracted using a bounding box so that it contained sufficient maxillary sinus segmentation for analysis. Finally, data augmentation was conducted for just the training dataset, using mirror images that were reversed left to right and rotated −30, −10, 10, and 30 degrees.

4.6 IMAGE LABELING AND DATASET DISTRIBUTIONS:

All subjects were independently labeled twice as "normal" or "sinusitis" by two radiologists. Labeling was first evaluated with the original images on a picture archiving communication system (PACS) and secondly with the resized images that were used for the actual learning data. Datasets were defined as the internal dataset and temporal dataset, with the temporal dataset used to evaluate the test. The internal dataset was randomly split into training (70%), validation (15%), and test (15%) subsets. The distribution of internal the test dataset consisted of 32% maxillary sinusitis, 32% Frontol sinusitis, and 34% Normal.

4.7 ACTIVATION FUNCTION

Activation function serves as a decision function and helps in learning of intricate patterns. The selection of an appropriate activation function can accelerate the learning process. In literature, different activation functions such as sigmoid, tanh, maxout, SWISH, ReLU, and variants of ReLU, such as leaky ReLU, ELU, and PReLU are used to inculcate non-linear combination of features

4.8 Operations using NumPy:

Using NumPy, a developer can perform the following operations:

• Mathematical and logical operations on arrays.

• Fourier transforms and routines for shape manipulation.

• Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index. Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

Data Type Objects (dtype)

A data type object describes interpretation of fixed block of memory corresponding to an array, depending on the following aspects:

• Type of data (integer, float or Python object)

• Size of data

• Byte order (little-endian or big-endian)

• In case of structured type, the names of fields, data type of each field and part of the memory block taken by each field

• If data type is a subarray, its shape and data type The byte order is decided by prefixing '<' or '>' to data type. '<' means that encoding is little endian (least significant is stored in smallest address)

4.9 TENSORFLOW:

Tensor Flow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using Tensor Flow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery.

Based on our experience with Disbelief and a more complete understanding of the desirable system properties and requirements for training and using neural networks, we have built Tensor Flow, our second-generation system for the implementation and deployment of large scale machine learning models. Tensor Flow takes computations described using a dataflow-like model and maps them onto a wide variety of different hardware platforms, ranging from running inference on mobile device platforms such as Android and iOS to modest sized training and inference systems using single machines containing one or many GPU cards to large-scale training systems running on hundreds of specialized machines with thousands of GPUs. Having a single system that can span such a broad range of platforms significantly simplifies the real-world use of machine learning system, as we have found that having separate systems for large-scale training and small-scale deployment leads to significant maintenance burdens and leaky abstractions. Tensor Flow computations are expressed as stateful dataflow graphs (described in more detail in Section 2), and we have focused on making the system both flexible enough for quickly experimenting with new models for research purposes and sufficiently high performance and robust for production training and deployment of machine learning models. For scaling neural network training to larger deployments, Tensor Flow allows clients to easily express various kinds of parallelism through replication and parallel execution of a core model dataflow graph, with many different computational devices all collaborating to update a set of shared parameters or other state. Modest changes in the description of the computation allow a wide variety of different approaches to parallelism to be achieved and tried with low effort. Some Tensor Flow uses allow some flexibility in terms of the consistency of parameter updates, and we can easily express and take advantage of these relaxed synchronization requirements in some of our larger deployments. Compared to Disbelief, Tensor Flow's programming model is more flexible, its performance is significantly better, and it supports training and using a broader range of models on a wider variety of heterogeneous hardware platforms.

In a Tensor Flow graph, each node has zero or more inputs and zero or more outputs, and represents the instantiation of an operation. Values that flow along normal edges in the graph (from outputs to inputs) are tensors, arbitrary dimensionality arrays where the underlying element type is specified or inferred at graph-construction time. Special edges, called control dependencies, can also exist in the graph: no data flows along such edges, but they indicate that the source node for the control dependence must finish executing before the destination node for the control dependence starts executing. Since our model includes mutable state, control dependencies can be used directly by clients to enforce happens before relationships. Our implementation also sometimes inserts control dependencies to enforce orderings between otherwise independent operations as a way of, for example, controlling the peak memory usage.

4.10 TENSORFLOW IMPLEMENTATION:

The main components in a Tensor Flow system are the client, which uses the Session interface to communicate with the master, and one or more worker processes, with each worker process responsible for arbitrating access to one or more computational devices (such as CPU cores or GPU cards) and for executing graph nodes on those devices as instructed by the master. We have both local and distributed implementations of the Tensor Flow interface. The local implementation is used when the client, the master, and the worker all run on a single machine in the context of a single operating system process (possibly with multiple devices, if for example, the machine has many GPU cards installed). The distributed implementation shares most of the code with the local implementation, but extends it with support for an environment where the client, the master, and the workers can all be in different processes on different machines. In our distributed environment, these different tasks are containers in jobs managed by a cluster scheduling system. These two different modes are illustrated. Most of the rest of this section discusses issues that are common to both implementations, while this discusses some issues that are particular to the distributed implementation.

Data Parallel Training- One simple technique for speeding up SGD is to parallelize the computation of the gradient for a mini-batch across mini-batch elements. For example, if we are using a mini-batch size of 1000 elements, we can use 10 replicas of the model to each compute the gradient for 100 elements, and then combine the gradients and apply updates to the parameters synchronously, in order to behave exactly as if we were running the sequential SGD algorithm with a batch size of 1000 elements. In this case, the Tensor Flow graph simply has many replicas of the portion of the graph that does the bulk of the model computation, and a single client thread drives the entire training loop for this large graph. The Tensor Flow system shares some design characteristics with its predecessor system, Disbelief, and with later systems with similar designs like Project Adam and the Parameter Server project. Like Disbelief and Project Adam, Tensor Flow allows computations to be spread out across many computational devices across many machines, and allows users to specify machine learning models using relatively high-level descriptions. Unlike Disbelief and Project Adam, though, the general-purpose dataflow graph model in Tensor Flow is more flexible and more amenable to expressing a wider variety of machine learning models and optimization algorithms.

4.11 LIMITATIONS:

The VGG-16 and VGG-19 architecture consist of large kernel-size filters with multiple 3×3 kernel-size filters, one after another. Within a given receptive field (the effective area size of an input image on which output depends), multiple stacked smaller sized kernels are better than a single larger sized kernel because multiple non-linear layers increase the depth of the network, enabling it to learn more complex features at a lower cost. As a result, the 3×3 kernels in the VGG architecture help to retain more fine details of an image. The ResNet architecture is similar to the VGG model, consisting mostly of 3×3 filters. Additionally, the ResNet model has a network depth of as large as 152.

Therefore, it achieves better accuracy than VGG and Google Net, while being computationally more efficient than VGG. While the VGG and ResNet models achieve phenomenal accuracy, their deployment on even the most modest sized GPUs is a problem because of the massive computational requirements, both in terms of memory and time. There are several limitations to this study. First, the external test dataset in multiple medical centers did not be included for reproducibility. In the case of X-ray equipment, there is a relatively small difference in performance compared to other medical imaging equipment, depending on the manufacturer or model.

In this study, therefore, the external test dataset in other medical centers did not be included. In the case of local medical center using relatively old equipment; however, an additional performance evaluation is also required to utilize artificial intelligence (AI) assistive software. Second, the proposed majority decision algorithm was optimized to evaluate only maxillary sinusitis. Therefore, there is a limitation to evaluate sinusitis in frontal, ethmoid, and sphenoid. In order to utilize AI based assistive software in the future, further study is underway because it is necessary to evaluate sinusitis at other locations as well as maxillary. Third, it lacks pattern recognition and representation methods that can solve black-box in deep learning. It needs to determine a reasonable consensus for solving the black-box problem. The feature recognition based activation map was used to solve the black-box problem in deep learning. As it can be shown from the results, not only classification but also lesion localization can be expressed as a result. It helps medical doctors make a reasonable inference about the deep learning analysis. However, it is not enough to understand all deep leaning procedures.

For example, it is difficult to understand the pattern of each learned CNN model. By understanding the pattern recognition capabilities of each model, we can understand the advantages and disadvantages of each model and achieve the optimization of the overall AI system. To overcome this limitation, a feature connectivity representation should be available for each layer to determine which feature weights are strong. In addition to feature representation, text-based description algorithm can be applied to overcome the black-box limitation in a medical application using the convolutional recurrent neural network (CRNN) that is the combination CNN and recurrent neural network (RNN) (20,21). A majority decision algorithm with multiple CNN models was shown to have high accuracy and significantly more accurate lesion detection ability compared to individual CNN models. The proposed deep learning method using PNS X-ray images can be used as an adjunct tool to help improve the diagnostic accuracy of maxillary sinusitis.

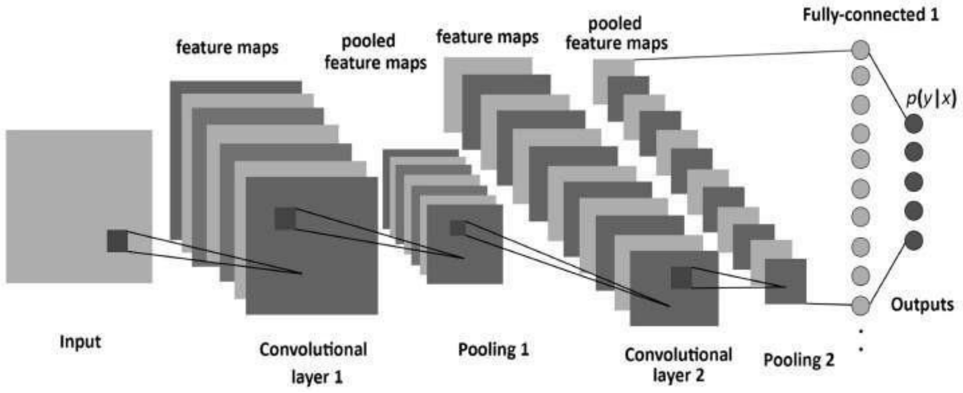


FIGURE 4.1: Pooling Step

If we wish to reduce the dimensionality of individual feature maps and yet sustain the crucial information, we use Spatial Pooling commonly known as down sampling or subsampling. This type of Pooling is of varied types: Average, Sum, Maximum, etc. For Max Pooling, we first specify a spatial neighborhood and then pick out the largest element of that feature rectified map within that window. Instead of largest element, if we pick the Average one it’s called Average Pooling and when the summation of elements in the window is taken, we call it Sum Pooling. Max Poolinghas proven itself as the best. The Figure 5.2 below shows a Pooling operation to every feature map separately and results 3 output maps from the 3 input maps.

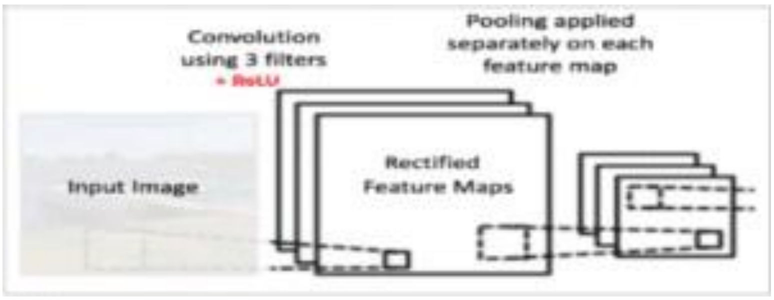

Convolution using 3 filters + ReLU
Pooling applied separately on each feature map
Input Image
Rectified Feature Maps

FIGURE 4.2: Pooling Operation

The basic purpose of pooling is to tremendously reduce the structural size of the input given. More specifically, the following are achieved through pooling:

- Genera.tes input representations i.e. feature dimensions that are small and so,more manageable.
- Lessens the parameters used and the computations in the network to efficiently control over fitting.
- Makes the network resistant to distortions, translations in the input image and to small transformations.
- Though minute distortions in the input shall not distort the pooling output as average/maximum value is used in the local neighborhood.
- Helps in arriving at an equi-variant representation of the input image. This is extremely useful as it enables us to detect small objects inside our image irrespective of where they are placed.

The working of Convolution, ReLU and Pooling till layers form the basic structure of any CNN. Generally, there are 2 sets of Convolution, ReLU and Pooling layers; convolution is performed on the output of the first Pooling layer by the second convolution layer employing six filters and so, producing six feature maps as well. ReLU layer is implemented on all these six feature maps individually. Having done this, we’re left with the task of Max Pooling, which we do on each of the six rectified feature maps separately. Coordination among these layers helps in the extraction of useful features from the images fed, then it adds non-linearity in the network and cuts down feature dimension thereby, making the features equivariant to scaleand translate. Input to the Fully Connected Layer is provided by the output of the second pooling layer.

4.12 Fully Connected Layer

This layer is described as a Multilayer Perceptron which utilizes a softmax activation function present in the output layer. The words “Fully Connected” in the fully connected layer indicate that every neuron in the next layer is connected to every individual neuron in the previous layer. Pooling and convolutional layer outputs depict high-resolution features of the input image. The fully connected layer on the basis of the training dataset utilizes these features for categorizing input images into different classes. A more general way of learning non-linear combinations of such features is by inserting a fully-connected layer instead of using classification. For the task of classification, features of the convolutional and pooling layers serve the purpose but using a combination of these is the best choice. Summation of all the output possibilities of the Fully Connected Layer comes as 1. To ensure this, weoperate the Softmax activation function in the output layer. This function works onany arbitrary real valued vector and transforms it into a vector valued between oneand zero, so as to acquire a sum of 1.

4.13 Softmax Function

Softmax function calculates the probabilities distribution of the event over ’n’ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

4.14 Object Detection

An image classification or image recognition model simply detect the probability ofan object in an image. In contrast to this, object localization refers to identifying the location of an object in the image. An object localization algorithm will output the coordinates of the location of an object with respect to the image. In computer vision, the most popular way to localize an object in an image is

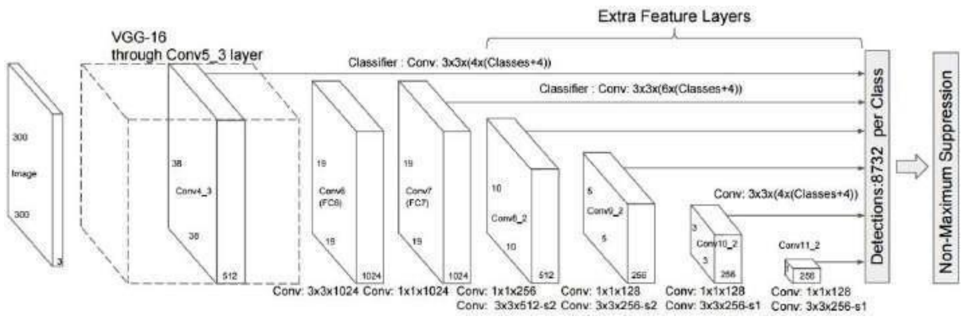


FIGURE 4.3: SSD Architecture

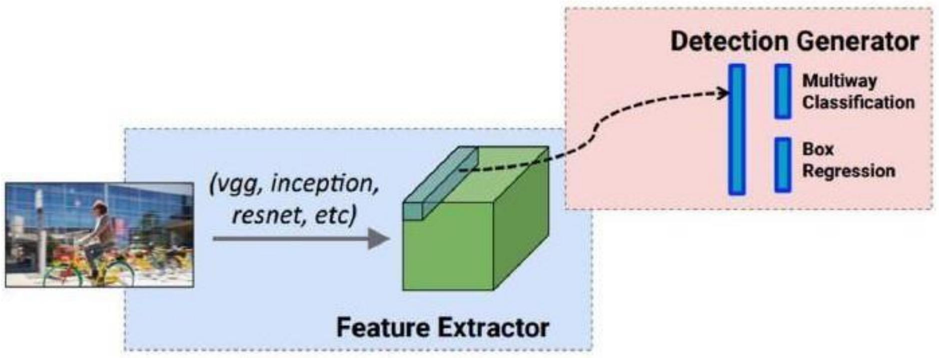


FIGURE 4.4: SSD Over all Idea

A discrete class and a continuous offset by which the anchor needs to be shifted to fit the ground-truth bounding box

During training SSD matches ground truth annotations with anchors. Each element of the feature map (cell) has a number of anchors associated with it. Any anchorwith an IoU (jaccard distance) greater than 0.5 is considered a match. Consider the case as shown in Figure 3.14, where the cat has two anchors matched and the dog has one anchor matched. Note that both have been matched on different feature maps

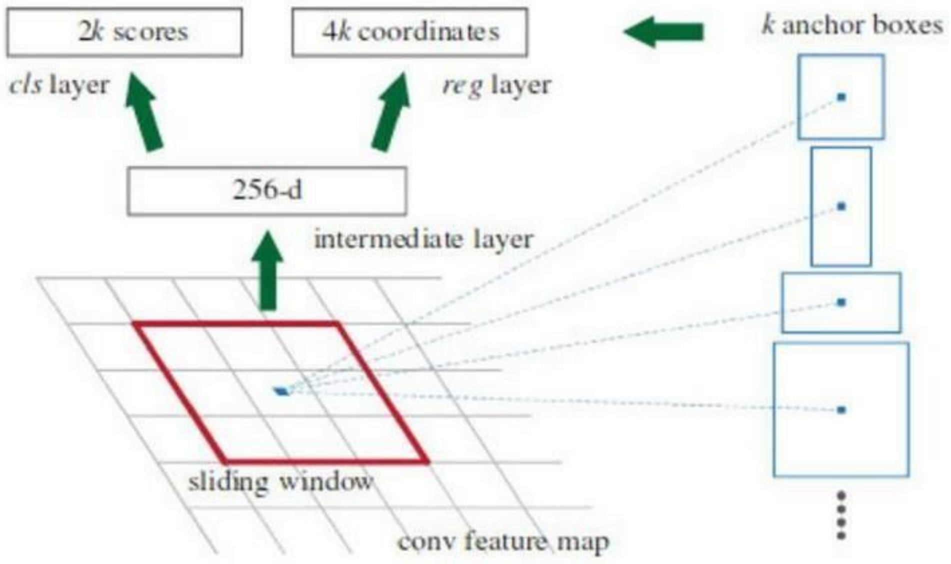


FIGURE 4.5: Anchors

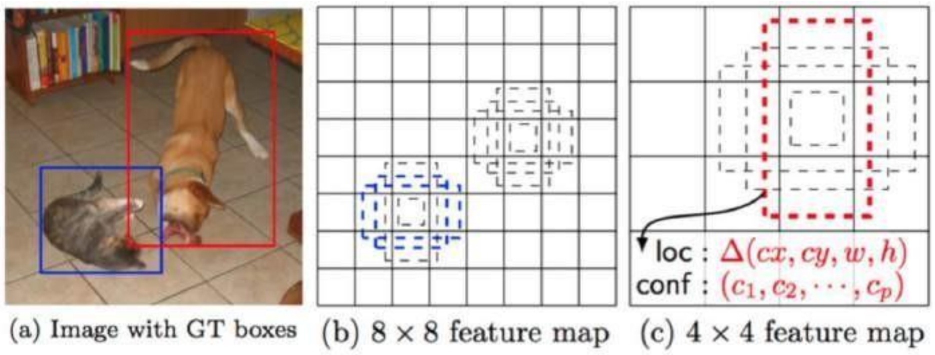

loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$
(a) Image with GT boxes
(b) 8×8 feature map
(c) 4×4 feature map

FIGURE 4.6: Matches

The loss function used is the multi-box classification and regression loss. The classification loss used is the softmax cross entropy and, for regression the smooth L1 loss is used. During prediction, non-maxima suppression is used to filter multiple boxes per object that may be matched as shown.

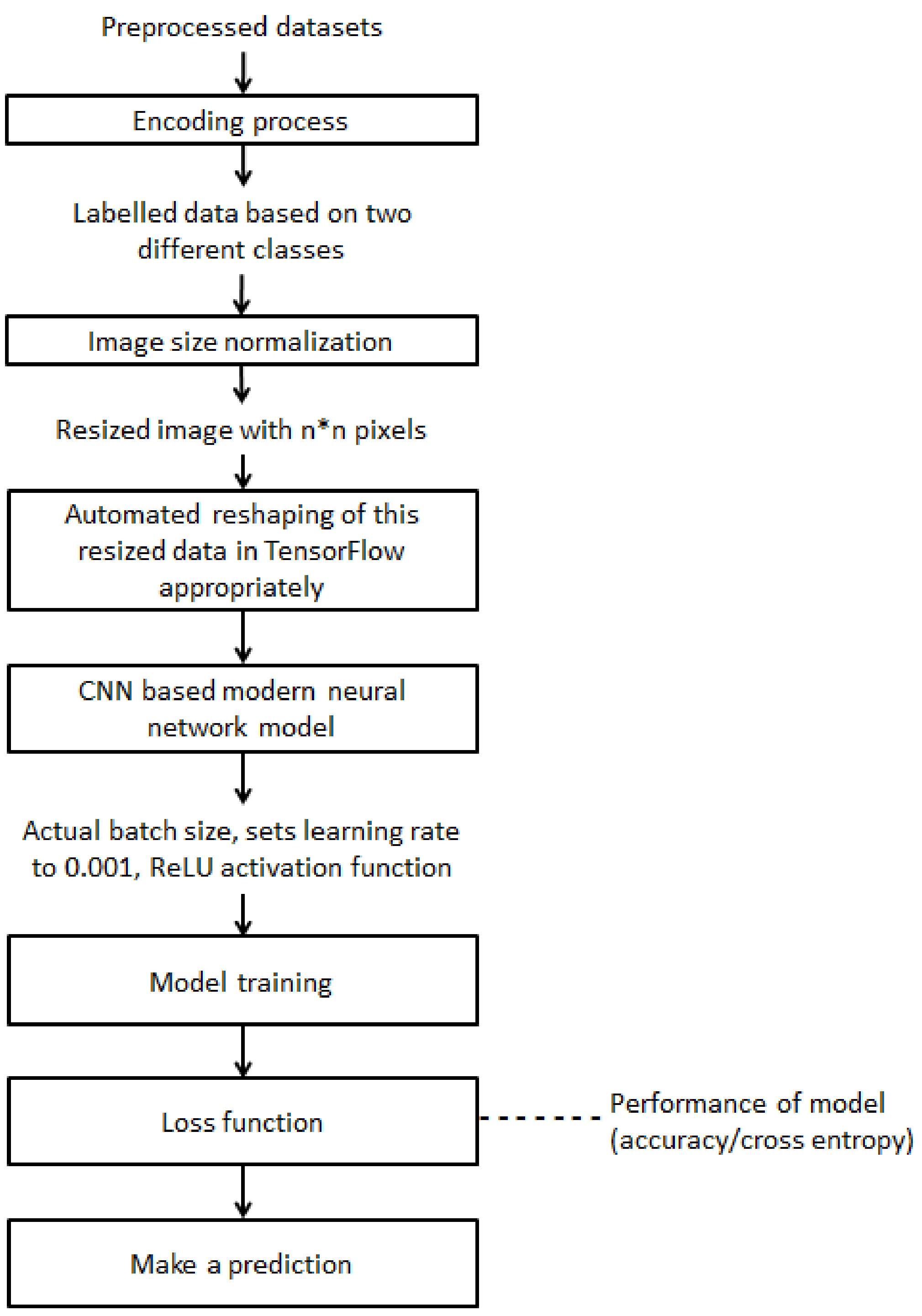

Preprocessed datasets
Encoding process
Labelled data based on two different classes
Image size normalization
Resized image with n*n pixels
Automated reshaping of this resized data in TensorFlow appropriately
CNN based modern neural network model
Actual batch size, sets learning rate to 0.001, ReLU activation function
Model training
Loss function
Performance of model (accuracy/cross entropy)
Make a prediction

FIGURE 4.7: Flow Chart of Classification Model

4.15 Labeling the image datasets

The MRI datasets are labeled based on two different classes. One class contains the normal Hand MRI images and the other class contains the OA affected Hand MRI images.

4.15.1 Image Transformation

All the MRI images of the datasets are in different pixel values. So, the image datasets with different pixel values are resized to n *n pixel values and made all images with similar pixel values.

TensorFlow based automated reshaping of data.

After the image transformation step, the data sets are appropriately reshaped for TensorFLow.

CNN based modern neural network model.

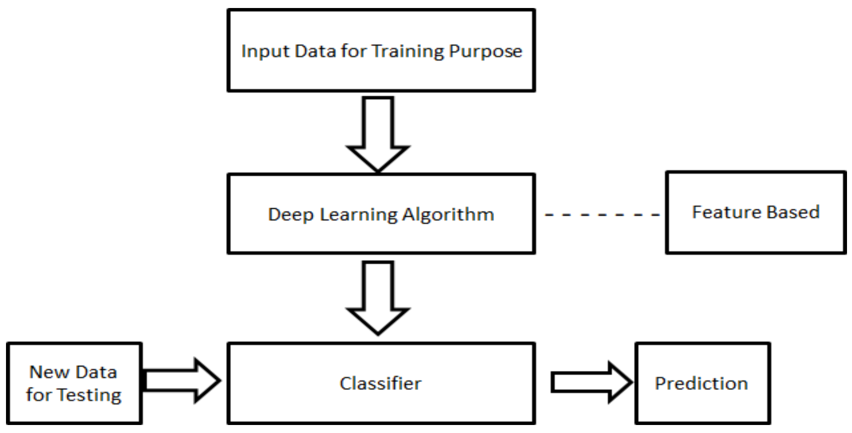


FIGURE 4.8: Deep Learning based Classification model

The key thing to understand while image classification in this project is that the model we are building is trained on two classes of normal Hand MRI and OA affected Hand MRI. The way we are going to achieve this classification by training an artificialneural network on image datasets and make the neural network learn to predict which class the input image belongs to, next time it sees an image the model will be able to predict if the input contains having a normal Hand MRI or a OA affected Hand MRI.

4.15.2 Activation Functions

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. We know, neural network has neurons that work in correspondence of weight, bias and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

Activation functions sit between layers in a network to introduce a non-linearity. Otherwise, the operations could be to a simple linear transformation and remove the benefits of building a deep model. The rectified linear unit (ReLU) is the default recommendation for an activation function in modern neural networks. It’s defined as the maximum of 0 and the input value and can be described as a nonlinear function made up of two linear pieces. Because of this, it preserves many of the properties that make models easy to optimize and generalize well on new data.

4.16 Batch Size

The number of random samples per training step is referred to as the batch size. In the past, it was believed that larger batches led to something called the generalization gap, where the accuracy of a model would drop if it was trained on unusually large batches. Recent work suggests other reasons for this decrease in accuracy. While common batch size ranges from 32 to 256. Using batches smaller than 32 samples can introduce a different kind of problem. Having too few data points that don’t represent the mean of the data well, may lead to slow and unstable training. Based on hardware limitations the largest possible batch sizes ranged from 24 to 48 samples depending on the current architecture. Whatever could be fit into memory was used for these experiments. Exceptions occurred when working with 2D convolutions in which case the batch size had to be limited to just 4 samples.

4.17 Data Set Collection

Some very large detection data sets, such as Pascal and COCO, exist already, but to train a custom object detection class, it is required to create and label own dataset.

For own data set, images of OA affected Hand MRI is collected from the website of National Institute of Health and also data provided by Invectus Innovation Pvt. Ltd., Noida. Only the OA affected Hand MRI images are used for this project. Ideally, got at least 300-400 OA Hand MRI images for this project. Then the data set splitted as 303 images for training; for the two possibilities of OA affected regions × namely, effusion and cartilage erosion (as per the medical expert of this area). Due to the limited amount of data, test files are splitted to 10%; of the total MRI images. For convenience, it is decided to resize all the images to 384 384 pixels before saving them so to create the required bounding boxes and not worry about having to resize the images down the line.

4.18 Creating Bounding Boxes

In order to train the object detection model, for each image the image’s width, height, and each class with their respective xmin, xmax, ymin and ymax bounding box is needed. For this project, the total number of MRI images labeled are 345 which include both training and test MRI images. Some of the MRIs from above mentioned labeled dataset with corresponding xml files showing details of their re- spective xmin, ymin, xmax and ymax bounding box values are shown by Table 4.2. Simply, bounding box is the frame that captures exactly where the possibilities ofOA affected area is in

the MRI image. Figure 4.8 is a labeled MRI image sample showing bounding boxes. These labels are created using LabelImg tool, an excel- lent open source free software that makes the labeling process much easier. It will save individual xml labels for each image, which we will convert into a csv table for training.
There are models in the Tensor Flow API which can be used depending on the needs. If a high-speed model that can work on detecting image feed at high fps is needed, the single shot detection (SSD) network works best. Some other object detection networks detect objects by sliding different sized boxes across the image and running the classifier many times on different sections of the image; this can be very resource consuming. As its name suggests, the SSD network determines all bounding box probabilities in one go; hence, it is a vastly faster model. However, with single shot detection, speed is gained at the cost of accuracy. For this project, single shot detection as the bounding box framework is used, but for the neural network architecture, the MobileNet model is used, which is designed to be used in mobile applications. The config file for SSD MobileNet has already configured and depending on the computer, the batch size in the config file is lowered to stop running out of memory.

4.19 Retrain the Model with Collected Data

Now trained the entire SSD MobileNet model has been trained on collected data from scratch which took a lot of training time. The easier solution is to take a model already trained on a large data set and clip off the last layer, which hasthe classes from the trained model, and replace it with required classes. By doing this, all the feature detectors trained in the previous model will be used and these features will be used to try to detect the new classes. Since only the last layer of thismobilenet model is retrained, a high-end GPU is not required (but it can certainly speed things up). Once the loss is consistently around the value of 1 or starts rising, TensorFlow training can be stopped by pressing ctrl+c. To train, simply run the ‘train.py‘ file in the object detection API directory.

Implementation of New Model With TensorFlow

Before starting an experiment with the newly trained model, exported the graphs for inference from TensorFlow. The latest ckpt from the data directory is used. Various graphs showing the development of classification loss, localization loss and total loss obtained while training the model is depicted in Figure.

After this, validated the model’s performance, in which those images are used which the model has never seen before; for validation images, some MRI images of different patient’s Hand MRI are used. About 10% of the total images are assigned to be validation images. It is to be noted that with a good number of validation images, multiple checkpoints can be tested to see which one performs best.

It is very clear to see that the very limited amount of data, about 100 images per class (for this case, effusion and cartilage erosion), was not enough to get a robust model. Though the detector was able to detect all the OA affected regions, but it was not able to detect the regions, may be due to absence of OA in Hand MRI or due to limited amount of data for this project. But, we believe this OA detection problem showcases the API’s capabilities well. It was the goal to gather all the steps to create such an object detection model which will be capable of detecting OA affected regions in Hand MRI images.

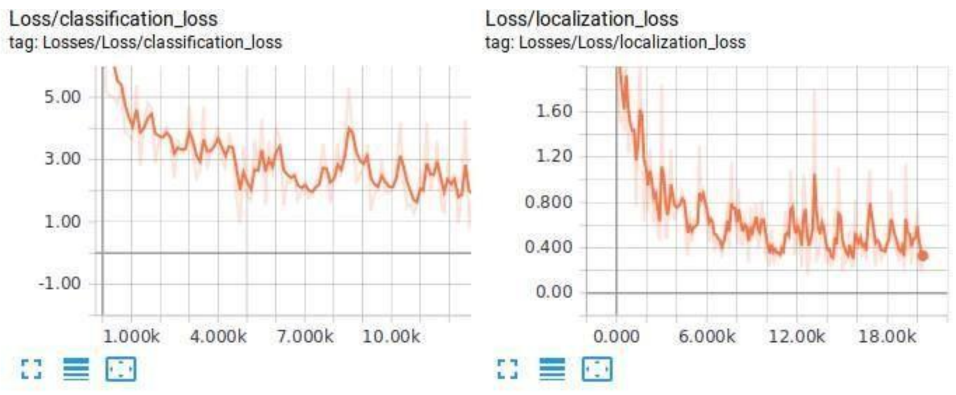


FIGURE 4.9: Graph showing development of classification loss and localization loss during training

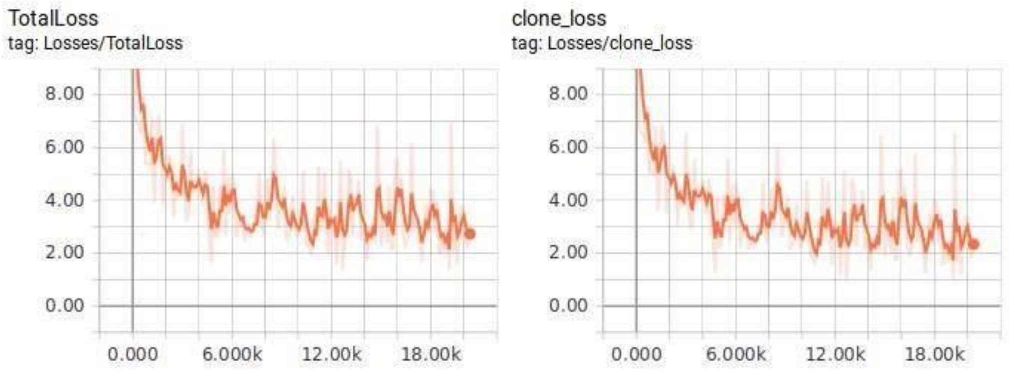


FIGURE 4.10: Graph showing of total loss during training our model

Object Detection with Customized Model

The pre-trained SSD model (ssd mobilenet v1 coco) was fine-tuned for the dataset using manually labeled images of OA Hand MRI. A provided configuration file (ssd mobilenet v1 coco.config) was used as a basis for the model configuration (after testing different configuration settings, the default values for configuration parame- ters were used). The provided checkpoint file for ssd mobilenet v1 coco was used as a starting point for the fine tuning process. The training was stopped after 12550time steps when the mean average precision (mAP) some what leveled out. The mAP values kept fluctuating even after increased value of mAP, which raised sus- picion that even longer training might improve the detection results. Training the model with CPU took a lot of time. The total loss value was reduced rapidly for this model due to starting from the pre-trained checkpoint file.

The fine-tuned model is fed to TensorFlow Object Detection API and tested on test data from OA Hand MRI images, and in addition on test data taken from different patients OA Hand MRI to see how model's specific factors affect the result. Themodel was then tested to see if the model specific fine-tuning improved the detection results.

Evaluation Metrics-The most relevant evaluation metrics for this application are precision (correctness),recall, sensitivity and mAP.

Precision is the ratio of the correctly positive labeled by our program to all positive labeled and describes how relevant the detection results are, Recall is the ratio of the correctly labeled by our program to all who are diabetic in reality and describes the percentage of relevant objects that are detected with the detector.

Generally, when precision increases, recall decreases and vice versa: a very sensitive model is able to catch large percentage of objects in an image, but it also generates high number of false positives, where as a model with high threshold for detection only produces few false positives but it also leaves a higher percentage of objects undetected. The right balance between these two depends on the application.

Sensitivity means percentage of diseased pixels properly excluded from the seg- mentation results out of all the pixels from the segmentation results outside of the ground truth disease

Mean Average Precision (mAP) is a commonly used metrics for comparing model performance in object detection. mAP summarizes the information in “precision-recall” curve to one number. Precision-recall curve is formed by sorting all the predicted bounding boxes (over all images in the evaluation set) assignedto a object class by their confidence rating, and then for each prediction, a recall value and a precision value are calculated. Recall is defined as proportion of TPs above the given rank out of all user tagged objects. Precision is defined as proportion of TPs in predictions above the given rank. From these precision-recall-pairs a precision-recall curve is formed. Average Precision (AP) is calculated as the “area under the precision-recall-curve”, i.e. it approximates precision averaged across all values of recall between 0 and 1 by summing precision values multiplied by the corresponding change in recall from one point in curve to the next. However, what VOC metrics call “AP” is actually interpolated average precision, which is defined “as the mean precision at a set of equally spaced recall levels”, the precision value corresponding to each recall interval being the maximum precision value observed in the curve within the interval. mAP of a model is then the mean of AP values ofall object classes. The interpolated AP is, of course, higher than the basic AP; this must be taken into consideration when comparing mAP values.

CHAPTER 5
SOFTWARE DESCRIPTION

5.1 PYTHON:

PYTHON 3.7:

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object- oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many a reason most platforms and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications. This tutorial introduces the reader informally to the basic concepts and features of the Python language and system.

It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off- line as well. For a description of standard objects and modules, see library-index. Reference-index gives a more formal definition of the language. To write extensions in C or C++, read extending-index and c-api-index. There are also several books covering Python in depth. This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most notes worthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in library-index. If you do much work on computers, eventually you find that there's some task you'd like to automate.

For example, you may wish to perform a search-and-replace over a large number of text files, or rename and rearrange a bunch of photo files in a complicated way. Perhaps you'd like to write a small custom database, or a specialized GUI application or a simple game. If you're a professional software developer, you may have to work with several C/C++/Java libraries but find the usual write/compile/test/re-compile cycle is too slow. Perhaps you're writing a test suite for such a library and find writing the testing code a tedious task. Or maybe you've written a program that could use an extension language, and you don't want to design and implement a whole new language for your application.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn't work, you can exit the interpreter by typing the following command: quit(). The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support read line. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line. The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file. A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell's -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote commands in its entirety with single quotes. Some Python modules are also useful as scripts. These can be invoked using python-m module [arg]..., which executes the source file for the module as if you had spelled out its full name on the command line. When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

There are tools which use doc strings to automatically produce online or printed documentation or to let the user interactively browse through code; it's good practice to include doc strings in code that you write, so make a habit of it. The execution of a function introduces a new symbol table used for the local variables of the function. More precisely, all variable assignments in a functions to read the value in the local symbol table; whereas variable references first look in the local symbol table, then in the local symbol tables of enclosing functions, then in the global symbol table, and finally in the table of built-in names. Thus, global variables cannot be directly assigned a value within a function (unless named in a global statement), although they may be referenced. The actual parameters (arguments) to a function call are introduced in the local symbol table of the called function when it is called; thus, arguments are passed using call by value (where the value is always an object reference, not the value of the object).1 When a function calls another function, a new local symbol table is created for that call. A function definition introduces the function name in the current symbol table. The value of the function name has a type that is recognized by the interpreter as a user-defined function. This value can be assigned to another name which can then also be used as a function.

Annotations are stored in the annotations attribute of the function as a dictionary and haven o effect on any other part of the function. Parameter annotations are defined by a colon after the parameter name, followed by an expression evaluating to the value of the annotation. Return annotations are defined by a literal ->, followed by an expression, between the parameter list and the colon denoting the end of the def statement.

The comparison operators in and not in, check whether a value occurs (does not occur) in a sequence. The operator is and does not compare whether two objects are really the same object; this only matters for mutable objects like lists. All comparison operators have the same priority, which is lower than that of all numerical operators. Comparisons can be chained. For example, a<b==c tests whether a is less than band moreover b equals c. Comparisons may be combined using the Boolean operators and the outcome of a comparison (or of any other Boolean expression) may be negated with not. These have lower priorities than comparison operators; between them, not has the highest priority and or the lowest, so that A and not B or C is equivalent to (A and (not B)) or C. As always, parentheses can be used to express the desired composition. The Boolean operators and are so-called short-circuit operators: their arguments are evaluated from left to right, and

evaluation stops as soon as the outcome is determined. For example, if A and C are true but Bis false, A and B and C does not evaluate the expression C. When used as a general value and not as a Boolean, the return value of a short-circuit operator is the last evaluated argument.

Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state. Compared with other programming languages, Python's class mechanism adds classes with a minimum of new syntax and semantics. It is a mixture of the class mechanisms found in C++ and Modula-3. Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name. Objects can contain arbitrary amounts and kinds of data. As is true for modules, classes partake of the dynamic nature of Python: they are created at runtime, and can be modified further after creation. In C++ terminology, normally class members (including the data members) are public (except see below Private Variables), and all member functions are virtual. A sin Modula-3, there are no short hands for referencing the object's members from its methods: the method function is declared with an explicit first argument representing the object, which is provided implicitly by the call. A sin Small talk, classes themselves are objects. This provides Semantics for importing and renaming. Unlike C++ and Modula-3, built-in types can be used as base classes for extension by the user. Also, like in C++, most built-in operators with special syntax (arithmetic operators, sub scripting etc.) can be redefined for class instances.(Lacking universally accepted terminology to talk about classes, I will make occasional use of Smalltalk and C++ terms. I would use Modula-3 terms, since its object- oriented semantics are closer to those of Python than C++, but I expect that few readers have heard of it.)

Objects have individuality, and multiple names (in multiple scopes) can be bound to the same object. This is known as aliasing in other languages. This is usually not appreciated on a first glance at Python, and can be safely ignored when dealing with immutable basic types (numbers, strings, tuples).However, aliasing has a possibly surprising effect on these mantic of Python code involving mutable objects such as lists, dictionaries, and most other types. This is usually used to the benefit of the program, since aliases behave like pointers in some respects. For example, passing an object is cheap since only a pointer is passed by the implementation; and if a function modifies an object passed as an argument, the caller will see the change — this eliminates the need for two different argument passing mechanisms as in Pascal.

A namespace is a mapping from names to objects. Most name spaces are currently implemented as Python dictionaries, but that's normally not noticeable in any way (except for performance), and it may change in the future. Examples of name spaces are: these to f built-in names (containing functions such as abs(), and built-in exception names); the global names in a module; and the local names in a function invocation. In a sense the set of attributes of an object also form a namespace. The important thing to know about namespaces is that there is absolutely no relation between names in different namespaces; for instance, two different modules may both define a function maximize without confusion — users of the modules must prefix it with the module name. By the way, I use the word attribute for any name following a dot — for example, in the expression z. real, real is an attribute of the object z. Strictly speaking, references to names in modules are attribute references: in the expression mod name. Func name, mod name is a module object and func name is an attribute of it. In this case there happens to be a straight forward mapping between the module's attributes and the global names defined in the module: they share the same namespace!1 Attributes may be read-only or writable. In the latter case, assignment to attributes is possible.

Module attributes are writable: you can write mod name. The_answer = 42. Writable attributes may also be deleted with the del statement. For example, del mod name .the_ answer will remove the attribute the_answer from the object named by mod name. Namespaces are created at different moments and have different lifetimes. The namespace containing the built-in names is created when the Python interpreter starts up, and is never deleted. The global namespace for a module is created when the module definition is read in; normally, module namespaces also last until the interpreter quits. The statements executed by the top-level invocation of the interpreter, either read from a script file or interactively, are considered part of a module called main, so they have their own global namespace.(The built-in names actually also live in a module; this is called built ins.) The local namespace for a function is created when the function is called, and deleted when the function returns or raises an exception that is not handled within the function. (Actually, forgetting would be a better way to describe what actually happens.) Of course, recursive invocations each have their own local namespace.

To speed uploading modules, Python caches the compiled version of each module in the pycache directory under the name module. version.pyc, where the version encodes the format of the compiled file; it generally contains the Python version number. For example, in CPython release 3.3 the compiled version of spam.py would be cached as pycache/spam.cpython-33.pyc. This naming convention allows compiled modules from different releases and different versions of Python to coexist. Python checks the modification date of the source against the compiled version to see if it's out of date and needs to be recompiled. This is a completely automatic process. Also, the compiled modules are platform-independent, so the same library can be shared among systems with different architectures. Python does not check the cache in two circumstances. First, it always recompiles and does not store the result for the module that's loaded directly from the command line. Second, it does not check the cache if there is no source module. To support anon-source (compiled only) distribution, the compiled module must be in the source directory, and there must not be a source module.

Some tips for experts:

You can use the -O or -OO switches on the Python command to reduce the size of a compiled module. The -O switch removes assert statements, the -OO switch removes both assert statements and doc strings. Since some programs may rely on having these available, you should only use this option if you know what you're doing. "Optimized" modules have an opt- tag and are usually smaller. Future releases may change the effects of optimization.

A program doesn't run any faster when it is read from a .pyc file than when it is read from a .py file; the only thing that's faster about .pyc files is the speed with which they are loaded.

The module compile all can create .pyc files for all modules in a directory.

There is more detail on this process, including a flow chart of the decisions

5.2 THONNY IDE:

Thonny is as mall and light weight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME, so Thonny only requires the GTK2 toolkit and therefore you only need the GTK2 runtime libraries installd to run it.

For compiling Thonny yourself, you will need the GTK (>= 2.6.0) libraries and header files. You will also need the Pango, Gliband ATK libraries and header files. Furthermore you need, of course, a C compiler and the Make tool; a C++ compiler is also required for the included Scintilla library. The GNU versions of these tools are recommended.

Compiling Thonny is quite easy. The following should do it:

```
% ./configure

% make

% make install
```

The configure script supports several common options, for a detailed list, type

% ./configure –help. There are also some compile time options which can be found in src/Thonny .h. Please see Appendix C for more information. In the case that your system lacks dynamic linking loader support, you probably want to pass the option --disable-vte to the configure script. This prevents compiling Thonny with dynamic linking loader support to automatically load libvte.so.4 if available. Thonny has been successfully compiled and tested under Debian 3.1 Sarge, Debian 4.0 Etch, Fedora Core 3/4/5, Linux From Scratch and FreeBSD 6.0. It also compiles under Microsoft Windows

At startup, Thonny loads all files from the last time Thonny was launched. You can disable this feature in the preferences dialog (see Figure 3-4). If you specify some files on the command line, only these files will be opened, but you can find the files from the last session in the file menu under the "Recent files" item. By default this contains the last 10 recently opened files. We can change the amount of recently opened files in the preferences dialog. We can start several instances of Thonny , but only the first will load files from the last session. To run a second instance of Thonny , do not specify any file names on the command-line, or disable opening files in a running instance using the appropriate command line option.

Thonny detects an already running instance of itself and opens files from the command-line in the already running instance. So, Thonny can be used to view and edit files by opening them from other programs such as a file manager. If you do not like this for some reason, you can disable using the first instance by using the appropriate command line option If you have installed libvte.so in your system, it is loaded automatically by Thonny, and you will have a terminal widget in the notebook at the bottom. If Thonny cannot find libvte.so at startup, the terminal widget will not be loaded. So there is no need to install the package containing this file in order to run Thonny . Additionally, you can disable the use of the terminal widget by command line option, for more information see Section3.2.You can use this terminal (from now on called VTE) nearly as a usual terminal program like xterm. There is basic clipboard support. You can paste the contents of the clipboard by pressing the right mouse button to open the popup menu and choosing Paste. To copy text from the VTE, just select the desired text and then press the right mouse button and choose Copy from the pop up menu. On systems running the X Window System you can paste the last selected text by pressing the middle mouse button in the VTE (on 2-button mice, the middle button can often be simulated by pressing both mouse buttons together).

As long as a project is open, the Make and Run commands will use the project’s settings, instead of the defaults. These will be used whichever document is currently displayed. The current project’s settings are saved when it is closed, or when Thonny is shut down. When restarting Thonny, the previously opened project file that was in use at the end of the last session will be reopened.

Execute will run the corresponding executable file, shell script or interpreted script in a terminal window. Note that the Terminal tool path must be correctly set in the Tools tab of the Preferences dialog - you can use any terminal program that runs a Bourne compatible shell and accept the "-e" command line argument to start a command. After your program or script has finished executing, you will be prompted to press the return key. This allows you to review any text output from the program before the terminal window is closed.

By default the Compile and Build commands invoke the compiler and linker with only the basic arguments needed by all programs. Using Set Includes and Arguments you can add any include paths and compile flags for the compiler, any library names and paths for the linker, and any arguments you want to use when running Execute.

Thonny has basic printing support. This means you can print a file by passing the filename of the current file to a command which actually prints the file.

However, the printed document contains no syntax highlighting.

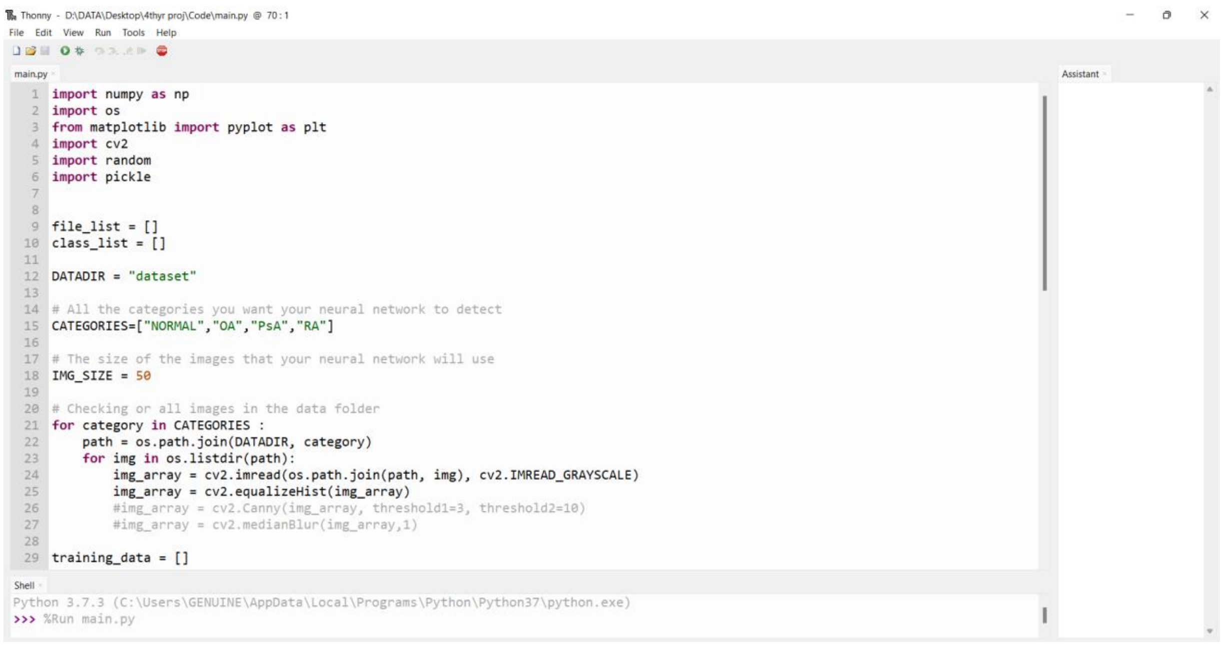
CHAPTER 6
RESULT

6.1 Result and Discussion

In this Project, we have assessed the efficiency of CNN-based finger joint detection. For this project, some X-ray pictures of the fingers are taken for training and testing. All images gives accurate results. The X-ray images are predicted for both healthy and unhealthy patients. The remedies are then expected to be displayed for patients who are affected by Arthritis. This will help doctors to see whether their prediction is right or not.

6.2 System output of hand arthritis

6.2.1 Simulation



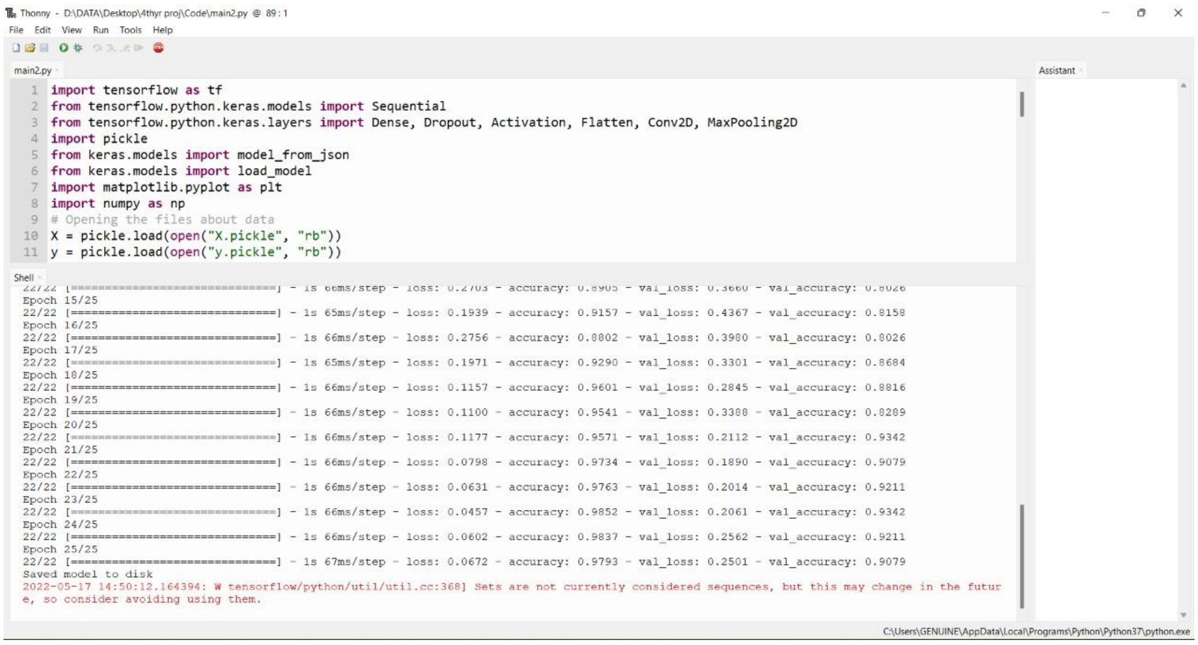
Thonny - D:\DATA\Desktop\4thyr proj\Code\main.py @ 70 : 1
File Edit View Run Tools Help
main.py
Assistant
Shell

```python
import numpy as np
import os
from matplotlib import pyplot as plt
import cv2
import random
import pickle


file_list = []
class_list = []

DATADIR = "dataset"

# All the categories you want your neural network to detect
CATEGORIES=["NORMAL","OA","PsA","RA"]

# The size of the images that your neural network will use
IMG_SIZE = 50

# Checking or all images in the data folder
for category in CATEGORIES :
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
        img_array = cv2.equalizeHist(img_array)
        #img_array = cv2.Canny(img_array, threshold1=3, threshold2=10)
        #img_array = cv2.medianBlur(img_array,1)

training_data = []
```

```
Python 3.7.3 (C:\Users\GENUINE\AppData\Local\Programs\Python\Python37\python.exe)
>>> %Run main.py
```

FIGURE 6.1: Simulation of hand arthritis detection

6.2.2 Performance

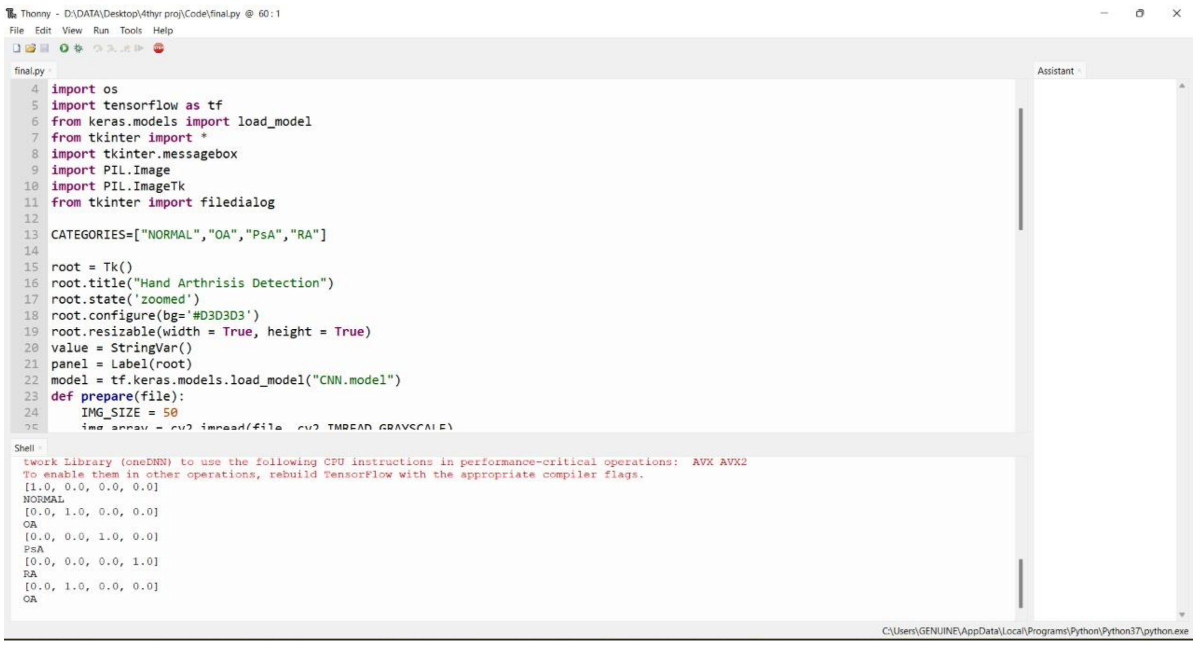

Thonny - D:\DATA\Desktop\4thyr proj\Code\main2.py @ 89 : 1
File Edit View Run Tools Help
main2.py
Assistant
Shell
C:\Users\GENUINE\AppData\Local\Programs\Python\Python37\python.exe

```python
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
import pickle
from keras.models import model_from_json
from keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np
# Opening the files about data
X = pickle.load(open("X.pickle", "rb"))
y = pickle.load(open("y.pickle", "rb"))
```

```
22/22 [==============================] - 1s 66ms/step - loss: 0.2703 - accuracy: 0.8905 - val_loss: 0.3660 - val_accuracy: 0.8026
Epoch 15/25
22/22 [==============================] - 1s 65ms/step - loss: 0.1939 - accuracy: 0.9157 - val_loss: 0.4367 - val_accuracy: 0.8158
Epoch 16/25
22/22 [==============================] - 1s 66ms/step - loss: 0.2756 - accuracy: 0.8802 - val_loss: 0.3980 - val_accuracy: 0.8026
Epoch 17/25
22/22 [==============================] - 1s 65ms/step - loss: 0.1971 - accuracy: 0.9290 - val_loss: 0.3301 - val_accuracy: 0.8684
Epoch 18/25
22/22 [==============================] - 1s 66ms/step - loss: 0.1157 - accuracy: 0.9601 - val_loss: 0.2845 - val_accuracy: 0.8816
Epoch 19/25
22/22 [==============================] - 1s 66ms/step - loss: 0.1100 - accuracy: 0.9541 - val_loss: 0.3388 - val_accuracy: 0.8289
Epoch 20/25
22/22 [==============================] - 1s 66ms/step - loss: 0.1177 - accuracy: 0.9571 - val_loss: 0.2112 - val_accuracy: 0.9342
Epoch 21/25
22/22 [==============================] - 1s 66ms/step - loss: 0.0798 - accuracy: 0.9734 - val_loss: 0.1890 - val_accuracy: 0.9079
Epoch 22/25
22/22 [==============================] - 1s 66ms/step - loss: 0.0631 - accuracy: 0.9763 - val_loss: 0.2014 - val_accuracy: 0.9211
Epoch 23/25
22/22 [==============================] - 1s 66ms/step - loss: 0.0457 - accuracy: 0.9852 - val_loss: 0.2061 - val_accuracy: 0.9342
Epoch 24/25
22/22 [==============================] - 1s 66ms/step - loss: 0.0602 - accuracy: 0.9837 - val_loss: 0.2562 - val_accuracy: 0.9211
Epoch 25/25
22/22 [==============================] - 1s 67ms/step - loss: 0.0672 - accuracy: 0.9793 - val_loss: 0.2501 - val_accuracy: 0.9079
Saved model to disk
2022-05-17 14:50:12.164394: W tensorflow/python/util/util.cc:368] Sets are not currently considered sequences, but this may change in the futur
e, so consider avoiding using them.
```

FIGURE 6.2: Epoch testing

Here 25 epochs(iterations) are performed to give the correct accuracy and loss percentage. According to the validation split epochs perform and give the level of accuracy and loss.


Thonny - D:\DATA\Desktop\4thyr proj\Code\final.py @ 60 : 1
File Edit View Run Tools Help
final.py
Assistant
Shell
C:\Users\GENUINE\AppData\Local\Programs\Python\Python37\python.exe

```python
import os
import tensorflow as tf
from keras.models import load_model
from tkinter import *
import tkinter.messagebox
import PIL.Image
import PIL.ImageTk
from tkinter import filedialog

CATEGORIES=["NORMAL","OA","PsA","RA"]

root = Tk()
root.title("Hand Arthrisis Detection")
root.state('zoomed')
root.configure(bg='#D3D3D3')
root.resizable(width = True, height = True)
value = StringVar()
panel = Label(root)
model = tf.keras.models.load_model("CNN.model")
def prepare(file):
    IMG_SIZE = 50
    img_array = cv2.imread(file, cv2.IMREAD_GRAYSCALE)
```

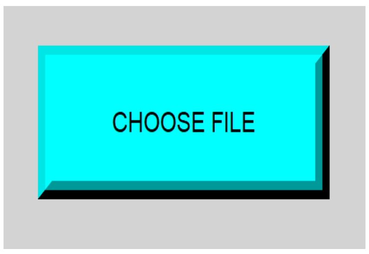
```
twork Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[1.0, 0.0, 0.0, 0.0]
NORMAL
[0.0, 1.0, 0.0, 0.0]
OA
[0.0, 0.0, 1.0, 0.0]
PsA
[0.0, 0.0, 0.0, 1.0]
RA
[0.0, 1.0, 0.0, 0.0]
OA
```

FIGURE 6.3: Detection output in the system

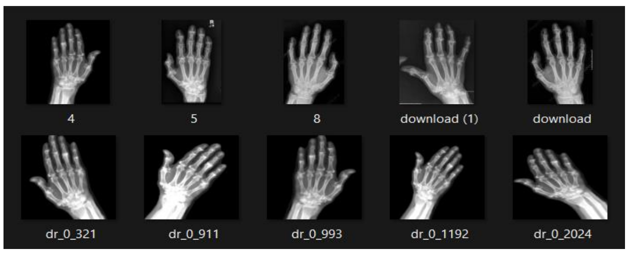
Here the array is 1×4 matrix with 4 columns of normal, OA, PSA, and RA. The value of 1 in particular column decides what kind of arthritis it is.

6.3 Application output

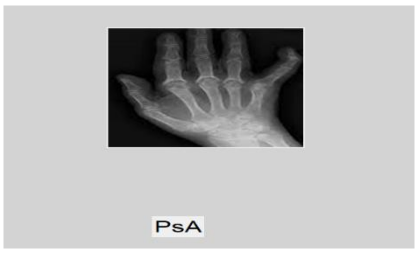
6.3.1 Detection of hand arthritis



Step 1



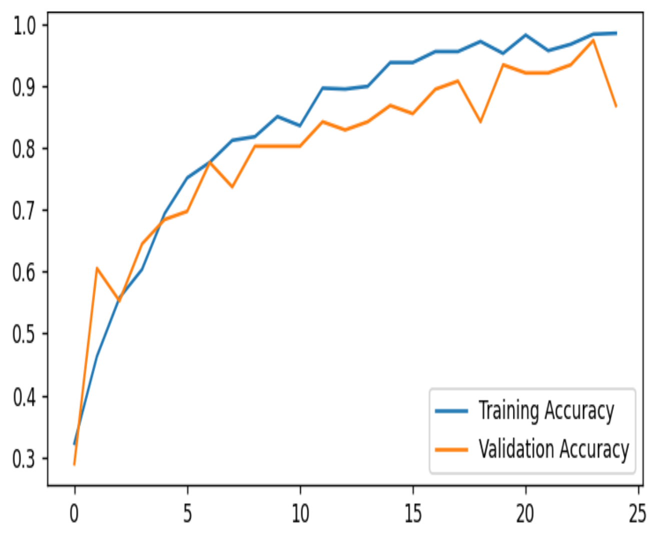
Step 2



Step 3

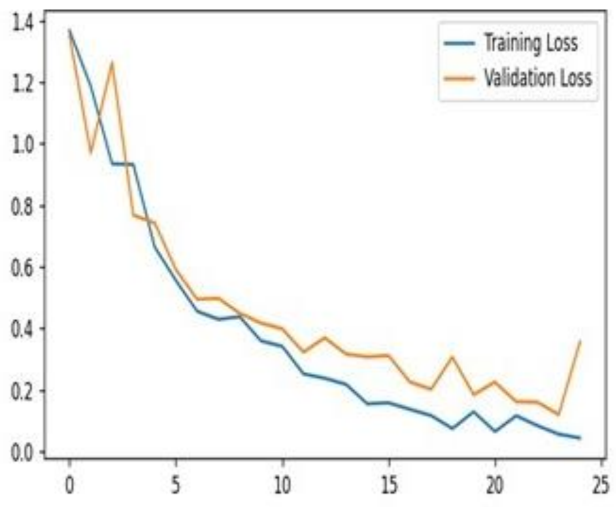
FIGURE 6.4: Result of hand arthritis detection

6.3.2 Graph Analysis



	Epoch(X-axis)	Accuracy(Y-axis)
Training	0	0.32
	3	0.6
	5	0.7
	10	0.82
	22	0.95
Validation	0	0.6
	7	0.79
	15	0.85
	20	0.9
	22	0.97

FIGURE 6.5: Training and validation accuracy



	Epoch(X-axis)	Loss(Y-axis)
Training	**0**	**1.38**
	7	**0.5**
	10	**0.2**
	18	**0.1**
	24	**0.09**
Validation	**0**	**1.38**
	3	**0.9**
	7	**0.4**
	15	**0.3**
	22	**0.2**

FIGURE 6.6: Training and validation loss

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

It was very interesting and adopting learning kind of experience to work with medical images in this project as it is playing a very big role in the medical field. This project took us through the various phases of project development and gave us the real insight into the world of machine learning techniques. The joy of working and the thrill involve while tackling the various problems and challenges gave us a feel of developers industry. It was due to this project we came to know how different machine learning techniques and methods can be designed in order to achieve a state of art solution to the problems like image classification and object detection. In this project, based on our created features from medical MRI images, later we were able to classify the normal Hand MRI images and the disease affected Hand MRI images. Also we were able to detect the actual location of the disease affected region in the Hand MRI images.

Some different adaptations, tests, and experiments have been left for the future due to lack of data and due to lack of time as well. The experiments require a large amount dataset to achieve better accuracy results. Also the experiments with real data are usually more time consuming, requiring even days to finish a single run. Future work for this project concerns with solving this object detection and localization problem with considering some more possibilities of OA cases like edema and osteophytes. The future scope of this project would be to use a large dataset of OA Hand MRIs and work on other two views such as axial and coronal. This would help this project in classifying and detecting the location of disease even in minor OA cases and with better accuracy. Other future scopes would also include the experiments tobe done on 3-D images, which may help to detect the presence disease region from whole Hand. Another future scope for this project would be to work on the medical images of other body joints also.

CHAPTER 8

REFERENCE

[1]. Detection of rheumatoid arthritis from hand radiographs using a convolutional neural network
Kemal Üreten, Hasan Erbay, Hadi Hakan Maraş, 2020
[2]. Bone erosion scoring for rheumatoid arthritis with deep convolutional neural networks
Janick Rohrbach, Tobias Reinhard, Beate Sick, Oliver Dürr
Computers & Electrical Engineering 2019
[3]. Automated Hand Osteoarthritis Classification Using Convolutional Neural Networks
Carmine Guida, Ming Zhang, Jordan Blackadar, Zilong Yang, Jeffrey B Driban, Jeffrey Duryea, Lena Schaefer, Charles B Eaton, Timothy McAlindon, Juan Shan
2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), 1487-1494, 2021
[4]. A deep neural network-based method for early detection of osteoarthritis using statistical data
Jihye Lim, Jungyoon Kim, Songhee Cheon
International journal of environmental research and public health 16 (7), 1281, 2019
[5]. Detection of Rheumatoid Arthritis Using a Convolutional Neural Network
AS Kumar, MS Mallikarjunaswamy, S Chandrashekara
International Conference on Innovative Computing and Communications, 607-618, 2022
[6]. Determining rheumatoid arthritis and osteoarthritis diseases with plain hand x-rays using convolutional neural network
Kemal Üreten, 2019
[7]. A Novel Approach for the Early Detection of Rheumatoid Arthritis on Hand and Wrist Using Convolutional Reinforcement Learning Techniques
KC Krishnachalitha, C Priya
Annals of the Romanian Society for Cell Biology, 3176–3186-3176–3186, 2021
[8]. A convolutional neural network with transfer learning for automatic discrimination between low and high-grade synovitis: A pilot study
Vincenzo Venerito, Orazio Angelini, Gerardo Cazzato, Giuseppe Lopalco, Eugenio Maiorano, Antonietta Cimmino, Florenzo Iannone
Internal and Emergency Medicine 16 (6), 1457-1465, 2021