Abstract

Multiple sclerosis (MS) is one of the most serious and widespread diseases, according to statistics published by the Atlas of (MS) and this is the open-source global compendium of data regarding the epidemiology of (MS). The number of people affected by multiple sclerosis has reached 2.8 million worldwide, and this number is rapidly increasing, and the early detection is Leading to successful treatment and reducing the progression of the disease significantly. MS appears in magnetic resonance imaging (MRI) of the brain or spinal cord and has various forms with white color called (MS lesions) and different levels of spread and appearance depending on the severity and progression of the disease depending on the condition of the patient.

In this project, we will contribute to the early detection of this disease by creating a system that utilizes deep learning techniques for classifying lesions in MS. This is achieved by inputting multiple images of Convolutional Neural Network (CNN) to extract features and classify images that contain lesions versus images that are healthy.

To evaluate the effectiveness of the system, we conducted experiments on a dataset Mendeley data website and the system achieves accuracy 74% in MS detection.

Our proposed automated Multiple sclerosis detection system has the potential to assist medical professionals in detecting cancerous tissues accurately and efficiently and this system we aim to enhance it to achieve higher accuracy than its current state.

Contents

Abstract 1

Chapter 1 6

Introduction 6

- **Multiple sclerosis (MS)** 7
- **Deep learning and image processing:** 8
- **System architecture:** 8
- **Aims of this project:** 8
- **Objectives:** 9
- **Scope of the project and future work:** 9
- **Suggested Solution:** 9
- **Suggested technologies:** 9
- **Deliverables** 9
- **Project Summary** 9
- **Project plan** 10

Chapter 2 11

Literature Review 11

- **Related Work** 12
  - **Project 1: Multiple Sclerosis lesions detection by a hybrid Watershed-Clustering algorithm (Lilla Bonanno, 2021)** 12
  - **Project 2: Brain Tumor Detection from MRI Images Using Optimization Segmentation Techniques (N Durga Indira, 2020)** 13
  - **Project 3: Brain tumor classification Mobile application:** 14
- **Comparing between this project and the related Work:** 17

Chapter 3 18

Requirements and analysis 18

- **Requirements specification** 19
  - **Functional requirements** 19
  - **Non-functional requirements** 19
  - **Software and hardware requirements** 19
- **Tools used to create the diagrams** 19
- **Initial system diagrams** 20
  - **Block Diagram** 20
  - **Flow Chart Diagram** 21
  - **Use case Diagram** 22

Class Diagram 23
Sequence Diagram 24
Discussion about the design 25
Code of ethics 25
Chapter 4 26
Design, Implementation, and testing 26
Classification System 27
What is the classification system? 27
Current life cycle 27
Problem Statement 29
Design Technique 30
Scale-Invariant Feature Transform (SIFT) 30
Convolutional Neural Networks (CNN) 31
Convolutional Neural Networks (CNN) with splitting data into train and test: 32
Convolutional Neural Networks (CNN) with splitting data into train, test, validation and adding data augmentation: 33
Comparison between all techniques 33
Implementation and result of SIFT: 33
Implementation and result of CNN with splitting data into training and testing sets: . 44
The summary of comparison 46
Implementation 47
Model Code: 47
Flutter Code: 60
Main File 60
Login page 61
Sign In page 68
Welcome Page 74
Scanning page 78
Check and upload page 80
Final Result page 85
Chapter 5 92
Results and Discussion 92
Results of 93
Model 93
Summary of model 93

Accuracy 94
Flutter App 94
- Login page 94
- Sign In page 96
- Check and Upload page 97
- Final Result page 98
Chapter 6 100
Future work and The conclusion 100
Future work 101
Conclusion 101
References 102

Figures:

Figure.1: There are 2.8 million people have MS in worldwide in 2020…………..……….9
Figure.2: Gantt Chart to Clarify the project timeline.……………………….………………12
Figure.3: steps to detect lesions of MS……………………………………………………...14
Figure.4: steps of prepare image and test in (PSO – DPSO – FO_DPSO) ……………....15
Figure.5: Mobile App for detecting brain……………………………………………..…...16
Figure.6: start app ……………………………………………………..……….........17
Figure.7: upload image…………………………………………………………….........17
Figure.8: result of test image that has “no tumor” ……………………………….............17
Figure.9: result of test image that has “tumor” ……………………………………..........18
Figure.10: Block Diagram…………………………………………………………………...22
Figure.11: Activity Diagram………………………………………………………………...23
Figure.12: Use case diagram………………………………………………………………...24
Figure.13: Class diagram…………………………………………………………………...25
Figure.14: Sequence diagram………………………………………………………………26
Figure.15: phases of Waterfall model (Pal, n.d) …………………………………………..30
Figure.16: phases of Agile model (Franciosi, 2020) ……………………………….……...31
Figure.17: sequence of SIFT algorithm implementation. ……………………………….....32
Figure.18: CNN architecture……………………………………………………………..34
Figure.19: SIFT: result of data shape and labels shape…………………………………...37
Figure.20: SIFT: result of paths_normal, paths_ms and labels………………………….…37
Figure.21: Result of SIFT in MS images……………………………………………….....42
Figure.22: features of MS images………………………………………………………...42
Figure.23: Result of SIFT in normal image………………………………………………...44
Figure.24: Features in normal image………………………………………………………44
Figure.25: Result of labels…………………………………………………………………...45
Figure.26: The result of SIFT algorithm with SVM……………………………………..…...46
Figure.27: Splitting data in CNN with splitting data into training and testing sets……..47
Figure.28: CNN training model after splitting data into train and test set………………47
Figure.29: Result of CNN training model after splitting data into train and test set……48
Figure.30: install Libraries……………………………………………………………..…...49
Figure.31: CV2 library version…………………………………………………………...49

Figure.32: output of data and labels shape……………………………………………...51
Figure.33: output 1 for data pre-processing………………………..…………………...52
Figure.34: output 2 for data preprocessing……………………………………………...52
Figure.35: output 3 for data pre-processing……………………………………………..52
Figure.36: output 4 for data pre-processing……………………………………………..53
Figure.37: Data visualizing output 1……………………………………………………..54
Figure.38: Data visualizing output 2…………………………..…………………...55
Figure.39: Data visualizing output 3…………………………...…………………...55
Figure.40: Data splitting output……………………………………………………….....56
Figure.41: Summary of CNN model……………………………………………………...59
Figure.42: Training the model output 1 ……………………………………………….....60
Figure.43: Training the model output2………………………...………………………...61
Figure.44: Screenshot of Login page from App…………………………….………...70
Figure.45: Screenshot of Sign in Page from app………………………………………...76
Figure.46: Screenshot of welcome page from app……………………………..……...80
Figure.47: Screenshot of scanning page from App……………………………………...82
Figure.48: Screenshot of Check and upload image………………………………………...87
Figure.49: Screenshot of final result page……………………………..……………...92
Figure.50: Prediction output 1…………………………………………….……………...93
Figure.51: Prediction output 2…………………………………………….……………...93
Figure.52: Summary of model with details…………………….………………………...95
Figure.53: Accuracy of CNN model……………………………………………………...96
Figure.54: Error message when email and password labels are empty. ……………..…...96
Figure.55: Screenshot of error message when email and password is not found……....97
Figure.56: Screenshot of create account in sign in page. ……………………………….98
Figure.57: Screenshot of image display in interface of app when click on the upload image button.…………………………………………………………………………....99
Figure.58: Screenshot of Test case 1: 'MS patient' image……………………..……...100
Figure.59: Screenshot of test case 2: 'Not MS patient'………………………………...101

Tables:

Table 1: The result of PSO, DPSO and FO-DPSO…………………………….……...16
Table 2: Clarify the difference between my project and other works ……………………19
Table 3: Summary of comparison between 3 techniques………………………………..48

Chapter 1

Introduction

Multiple sclerosis (MS)

Multiple sclerosis (MS) is the most prevalent disease at this time. MS is the disease that affects the nervous system in the human body, as the immune system produces white blood cells (WBC), which protect the body from antibodies and viruses that may lead to infection of the body with some diseases, but the defect occurs when WBCs begin to attack the nervous system and nerve cells In the brain and spinal cord, this attack causes the erosion of myelin, which is responsible for protecting and covering nerve cells.

When occurring erosion of nerve cells, the patient faces many problems in movement and imbalance during walking and some disturbances in vision and behavior and these symptoms vary according to the location of attacks in brain and spinal cord. According to the latest statistics published in 2020 from National Multiple Sclerosis Society show that more than 2.8 million people infected with MS and nearly 56.671 people have MS in Egypt (Clare Walton, Rachel King, et al, 2020). And the number of injuries is increasing dramatically as the number of injuries 2·2 million cases of multiple sclerosis worldwide in 2016, in Egypt was 29566 people and the number of deaths was changed and increased every day (Wallin, M.T., Culpepper, W.J., et al, 2019).

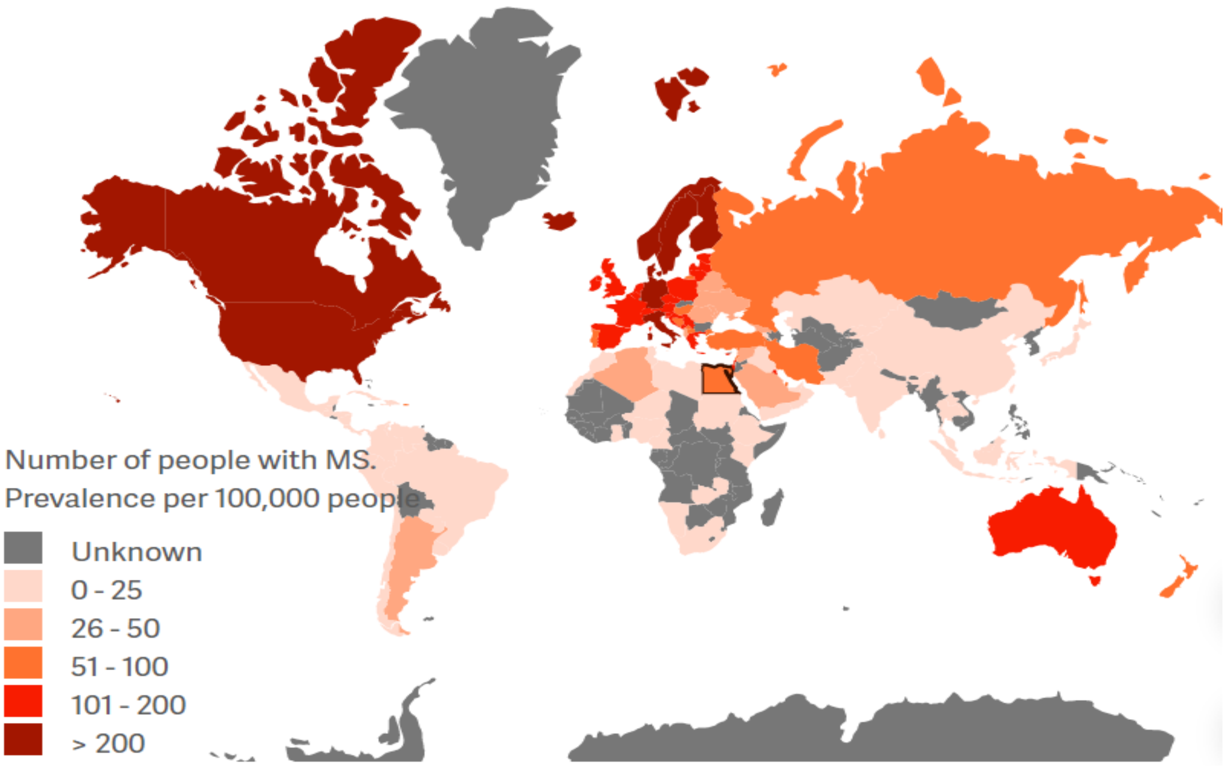


Figure.1: There are 2.8 million people have MS in worldwide in 2020(sharawy, 2020)

Experiment was done on 3795 patients and separate them into two groups, group A (are 2316 patients) that they have been diagnosed with MS patients before 2 years started appeared symptoms and group B (are 1479 patients) that they have been diagnosed with MS but from after more 2 years or 8 years from the start of appeared symptoms.

The group A, whose disease was discovered early had 6 times better results than the group B who received the same treatment, but their cases were diagnosed late, the risk cases in the group B increased by 42% than the group A.
And this experiment confirms the early detection is the most necessary for MS patients or for any disease (Chalmer, T.A., Baggesen, L.M. et al, 2018).

Deep learning and image processing:

Nowadays, image processing is the usage of digital computers or phones to process photos and enhance it if it request that, and in computer vision field include the image feature detectors and descriptors and this important algorithms in their fields and this allow to detect objects by edges or can say in general, image features can be categorized as edges, corners blobs, and ridges and so first we can use Scale Invariant Feature Transform (SIFT) (tabmir, 2022) technique to help us to detect MS lesions, SIFT transforms an image into a large collection of local feature vectors, each of which is invariant to image translation, scaling, and rotation, and partially invariant to illumination changes and affine or 3D projection.
Next step put result of SIFT technique in machine learning algorithm, in our case we can use supervised machine learning algorithms such as support vector machine (SVM) (alokesh985, 2022) to classify result if it MS category or health category 'no MS', SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line. In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data.
Deep learning and image processing have revolutionized the healthcare industry by providing advanced solutions for medical image analysis and diagnosis. Deep learning models, specifically convolutional neural networks (CNNs), have shown exceptional performance in detecting and classifying abnormalities in medical images. These models can learn and extract features from images faster and more accurate and this is what will be explained in this project.

System architecture:

The system architecture that the system of project will follow, Allow the user to take a picture or upload photo of Brain MRI then he will click to allow mobile application to access the photo to start the detection of lesions, classify it if it is MS or not and show the result of the report with a detailed description of the patient's condition such as (The type of MS- what is the symptoms of this type - how to handle them)

Aims of this project:

1- Provide an easier way for the doctors to be able to detect the MS diseases easier and more accurate.
2- help patients to detect the disease early, that will reduce the disease severity and progression.

3- Reaching high accuracy detecting MS.

Objectives:

Reducing the death rate due to this disease and limiting the number of people at risk or death, as it will contribute greatly to helping to detect the disease early.

Scope of the project and future work:

The project will start with only detecting the Lesions of MS from brain MRI and in the future application has ability to predict the degree of severity of the disease and to what extent will this patient's case develop? this is the best way to find the appropriate medicine and reduce the risk of this disease?

Suggested Solution:

As it is hard to detect the MS by traditional ways, I suggest using the detection techniques with Machine learning algorithm to allow the user (patient) investigate if he/she has MS or not.

Suggested technologies:

- Using Convolutional neural network(CNN) to detect MS lesions and classify image if contain MS or not.
- Framework: Flutter to build mobile application.
- Programming language: python, dart for mobile application.

Deliverables

It will appear to the user (the patient) whether he is a patient with multiple sclerosis or not, that show the result as a report has the information of patient and details of the detection result.

Project Summary

Chapter 1 – Introduction: This chapter discusses the project background and general information about the project.

Chapter 2 - Literature Review: This chapter discusses the previous related work done by other researchers or entities.

Chapter 3 – Requirements and analysis: This chapter provides the main software and hardware requirements and an initial design using UML diagrams.

Chapter 4 - Design, Implementation, and testing: This chapter will discuss the design approaches, and which one will be followed, and it will provide the implementation and testing for the project.

Chapter 5 - Results and discussion: This chapter will provide the findings and the achievements of the project, and it will also discuss the further work that can be investigated.

Chapter 6 – Conclusions: This should be a short chapter that conclude the project and the results especially in the previous chapter Results and discussion.

Project plan

Gantt Chart:

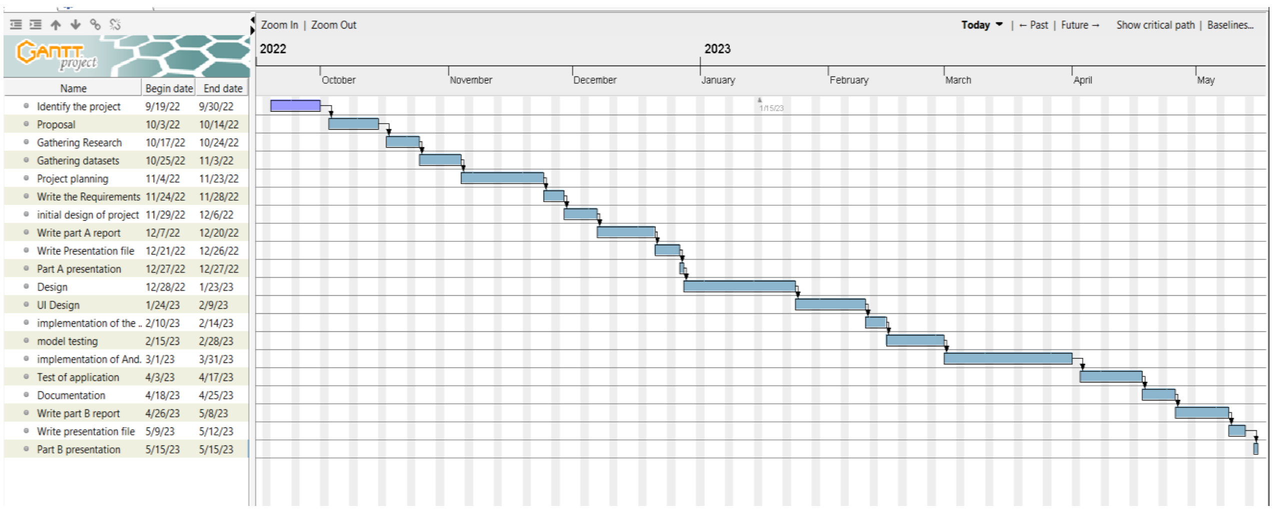


Figure.2: Gantt Chart to Clarify the project timeline.

Chapter 2

Literature Review

Related Work

Project 1: Multiple Sclerosis lesions detection by a hybrid Watershed-Clustering algorithm (Lilla Bonanno, 2021)

Year of Publication: 2021

- Using Computer Aided Diagnosis (CAD) system to detection Lesions of Multiple sclerosis (MS) in Brain magnetic resonance imaging (MRI) and this system based on hybrid watershed- clustering algorithm.
- Using a dataset of 20 patients and separating this dataset to two folder first folder has T1 images and second file has T2 images (T1, T2 are technique of MRI Scan).
- after doing the pre-processing of this dataset to make size, colour and resolution are fitting, then apply the model that using watershed-clustering algorithm to detect the lesions of Brain MRI to know if patient has MS or not.
- Result: Model separates 'true positive' if the result from CAD System say this image has

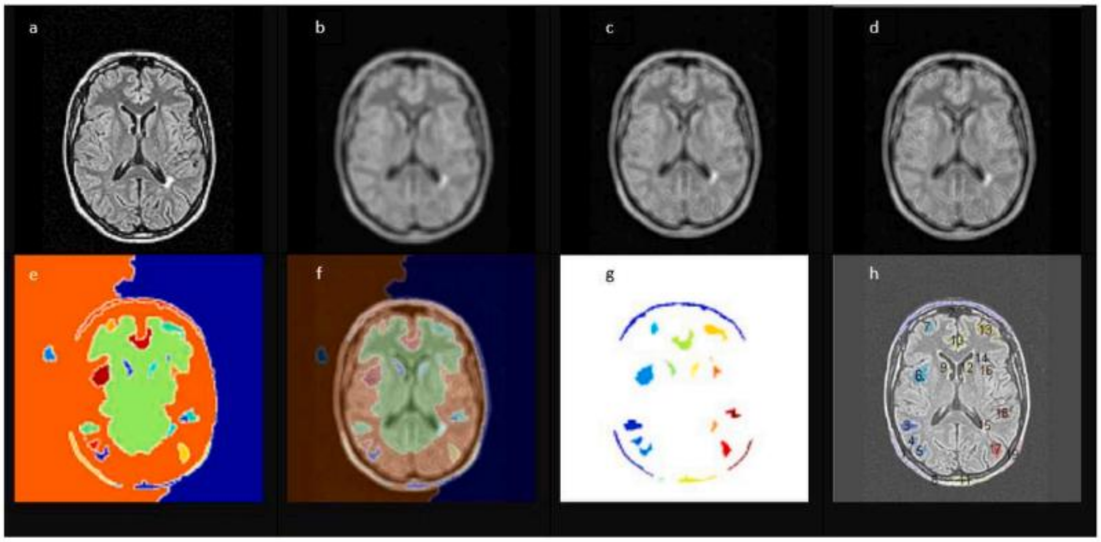


Figure.3: steps to detect lesions of MS

- lesion and result of algorithm tells this is correct, 'false positive' if the result from CAD

System say this image has not lesion but result of algorithm tells this is image has one or more lesions , 'true negative ' if the result from CAD System say this image has not lesion and result of algorithm tells this is correct, 'false negative' if the result from CAD System say this image has not lesion but result of algorithm tells this is part has new lesion that CAD doesn't know it.

Accuracy= 87%, sensitivity= 77% and specificity = 87%

Project 2: Brain Tumor Detection from MRI Images Using Optimization Segmentation Techniques (N Durga Indira, 2020)

Year of Publication: 2020

- To do segmentation of images or dataset this is the important step in any program to make detection step easier and more accurate, in this research apply 3 techniques (Particle Swam Optimization (PSO) (Khanal, 2020) segmentation technique and Darwinian Particle Swam Optimization (DPSO) (Micael S. Couceiro, Rui P. Rocha et al , 2021) and Functional Order Darwinian Particle Swam Optimization (FO-DPSO)) do comparison between them and find the method to do segmentation, the flow of program in figure.4.

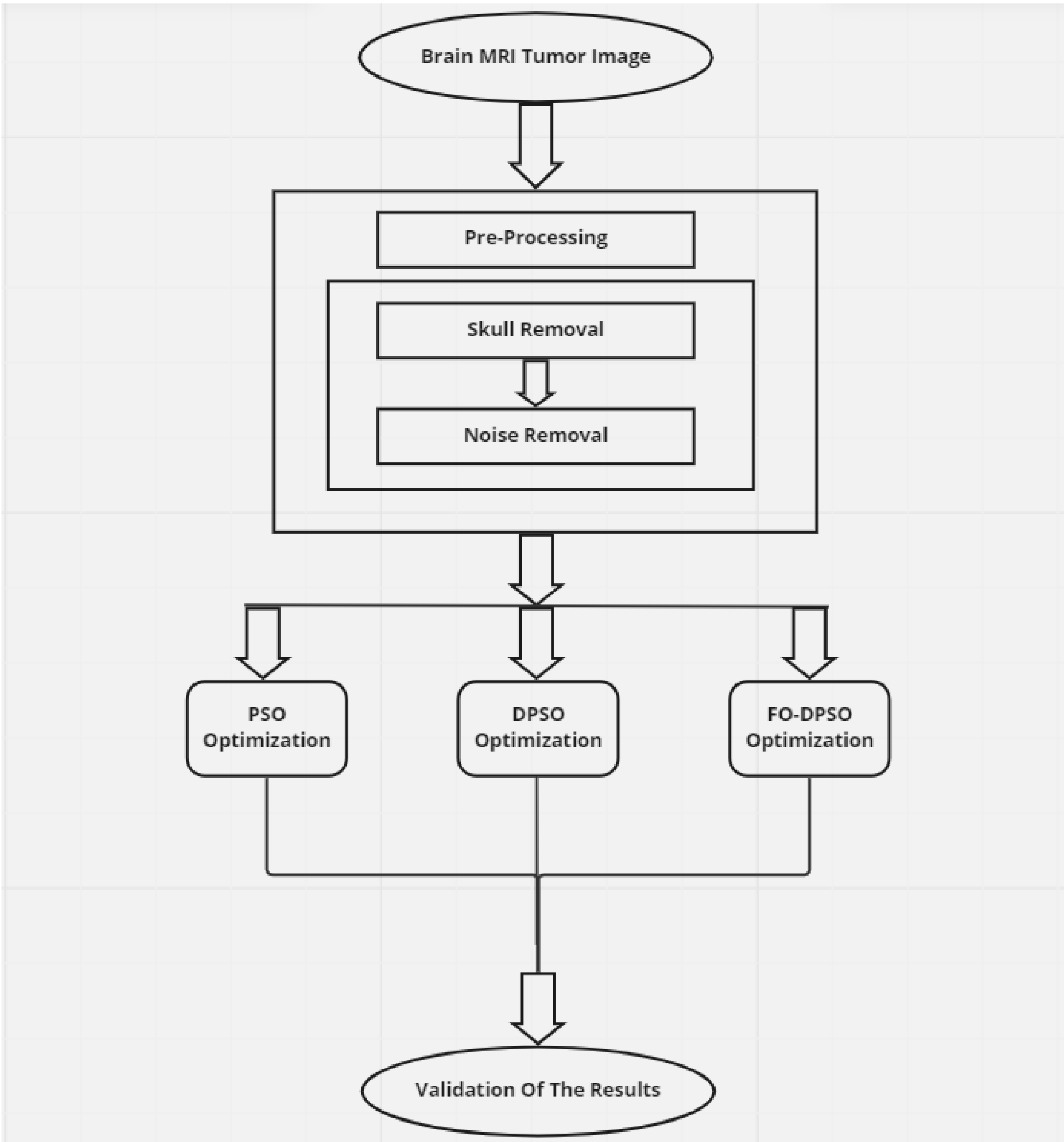


Figure.4: steps of prepare image and test in (PSO – DPSO – FO_DPSO)

- Result: FO-DPSO provides the best performance in segmentation images, figure.5 clarify that.

Parameter	**PSO**	**DPSO**	**FO-DPSO**
Iterations	10	10	10
Best Cost	1.2	0.86	0.83
Time (s)	2.95	2.5	1.4

Table 1: The result of PSO, DPSO and FO-DPSO

Project 3: Brain tumor classification Mobile application:

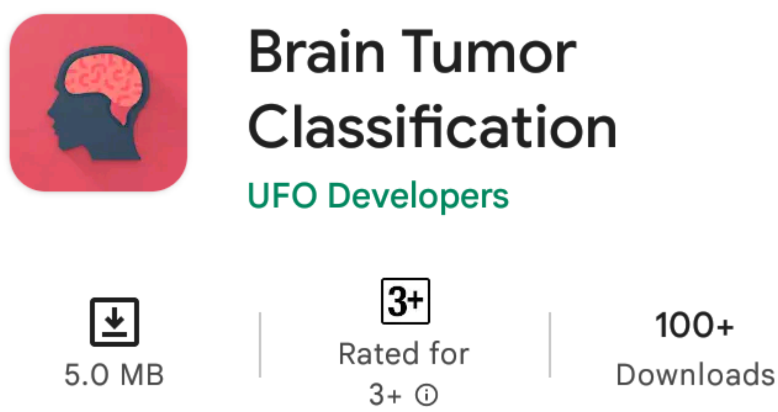


Figure.5: Mobile App for detecting brain

Brain tumor classification app is an application for detecting brain tumours from (MRI) using deep learning techniques and convolution neural network (CNN) (Wadhah Ayadi, Wajdi Elhamzi et al, 2021). It's available on both android app store and IOS app store. This application detects the brain tumor of and shows if user has brain tumor or not or if he/she at risk of contracting this disease or not, but this application cannot detect another disease such as MS.

<u>Screenshots from this app:</u>

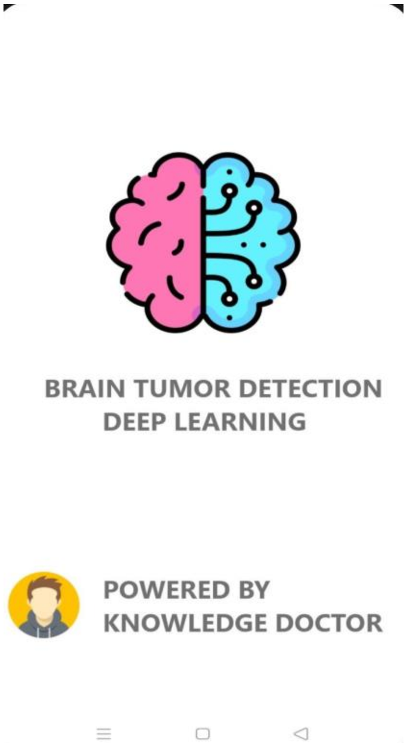


Figure.6: start app

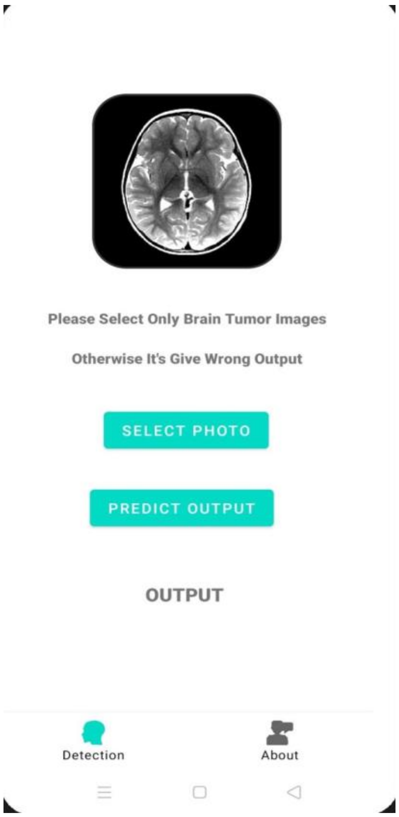


Figure.7: upload image

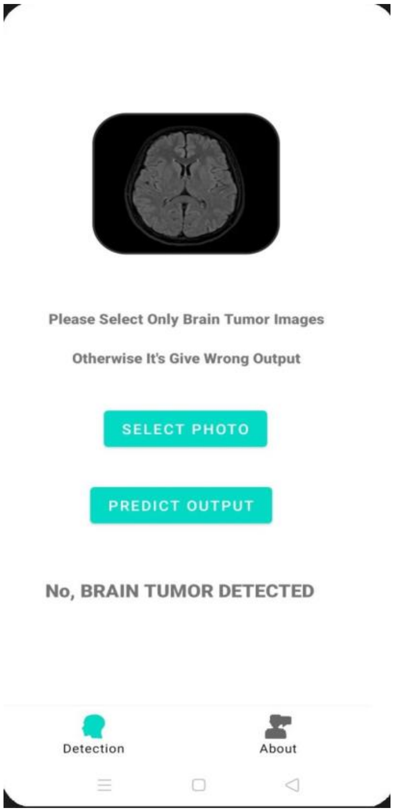


Figure.8: result of test image that has “no tumor”

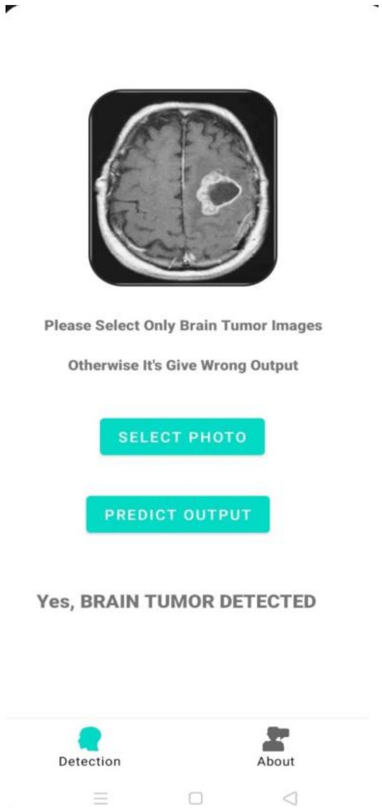


Figure.9: result of test image that has "tumor"

Comparing between this project and the related Work:

	My Project	Project 1: Multiple Sclerosis lesions detection by a hybrid Watershed-Clustering algorithm	Project 2: Brain Tumor Detection from MRI Images Using Optimization Segmentation Techniques	Project 3: Brain tumor classification Mobile application
Main Idea	Detecting MS lesions	Detecting MS lesions	Detecting Brain Tumor	Detecting Brain Tumor
Implemented as	Mobile app	Research paper	Research Paper	Mobile app
Accuracy	77%	77%	The optimization technique (Best cost =0.83, Time=1.4s)	Unknown
Mobile app Rating	Still unknown	....	....	3.5
Availability On android	Yes	....	....	Yes
Availability on IOS	No	....	....	No

Table 2: clarify the difference between my project and other works

My project is much similar to “Brain Tumor Classification Mobile Application” because both of them focus on detecting the disease, but my project focuses on detection of MS Lesions not brain tumor as this app.

Chapter 3

Requirements and analysis

Requirements specification

Functional requirements

1- The system should allow the user to signup / login. (Business requirement)
2- The application should allow the user to enter an image of MRI on the system. (Business requirement)
3- The application should allow the user to upload images of their MRI images. (Technical requirement)
4- The application should allow the user to take pictures (MRI image) from the camera. (Technical requirement)
5- The application should detect if this image has lesions or not. (Business requirement)
6- The application should classify and detect the disease if there is any. (Business requirement)
7- The application should provide information about the disease. (Business requirement)
8- The application should save the past MRI images and any previous information related to the patient. (Business requirement)
9- The application should display the result of detection as a report has the details of patient and if he/she has MS or not (Technical requirement)

Non-functional requirements

1- The application is clear and simple for any user and can use it without facing any problem. (Usability)
2- The system protects all information of users and prevents anyone to access or use it. (Security)

Software and hardware requirements

1-The operating system should be android 5.0 and up
2- The android device should have a camera with at least 8 megapixels.

Tools used to create the diagrams

All the provided diagrams are made using Miro online white board provide features to design the diagrams.

Initial system diagrams

Block Diagram

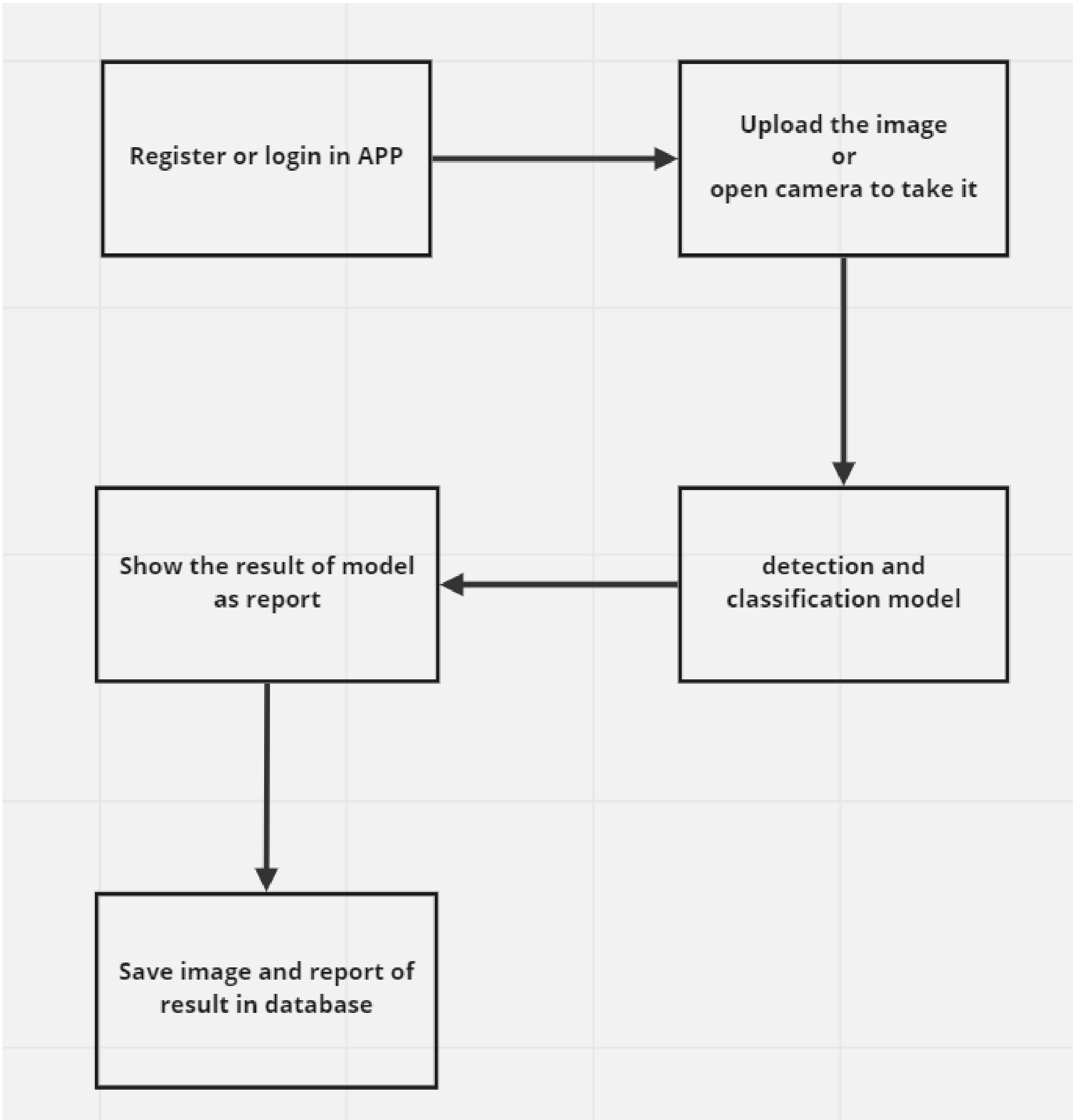


Figure.10: Block Diagram

Flow Chart Diagram

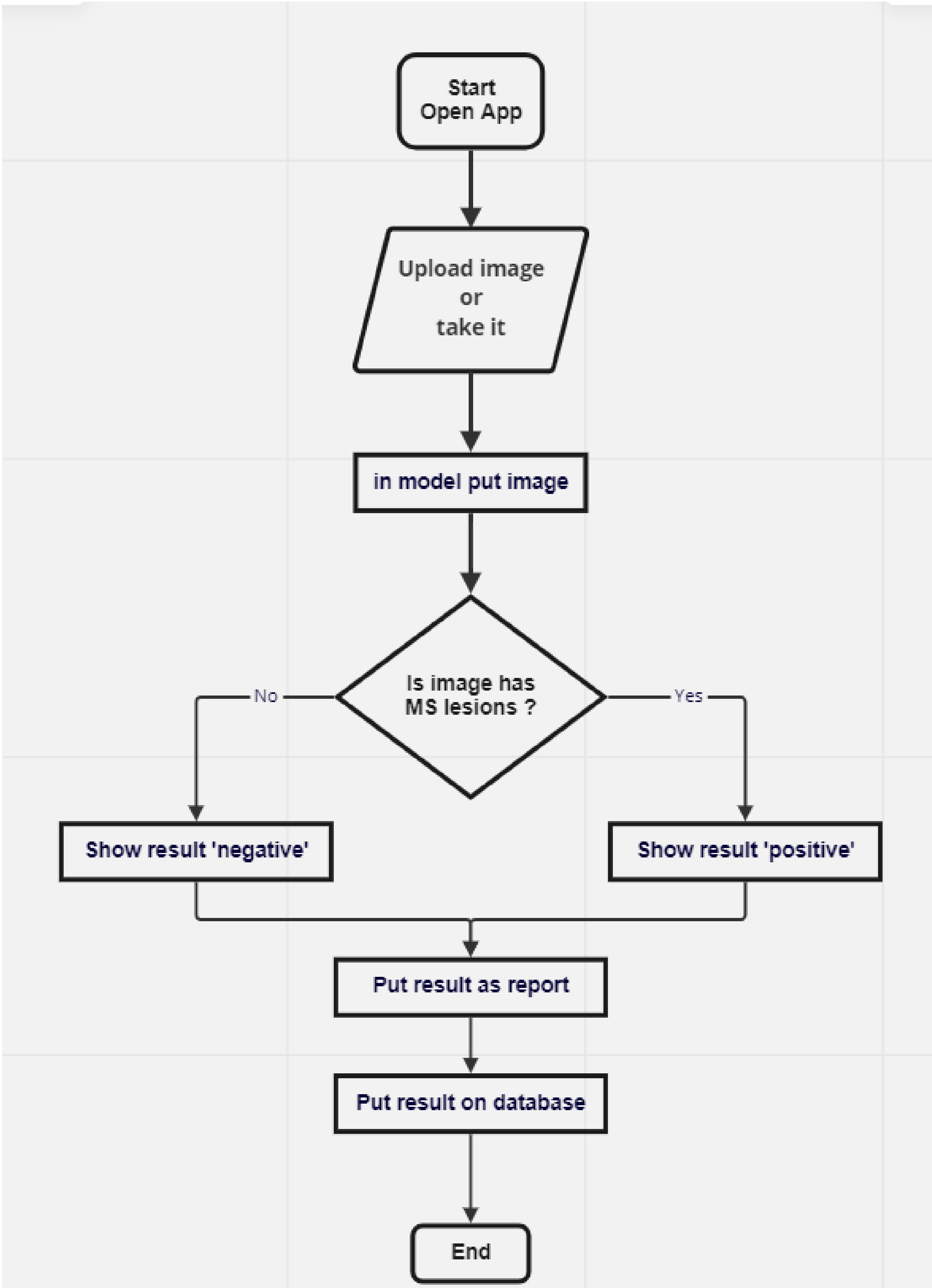


Figure.11: Activity Diagram

Use case Diagram

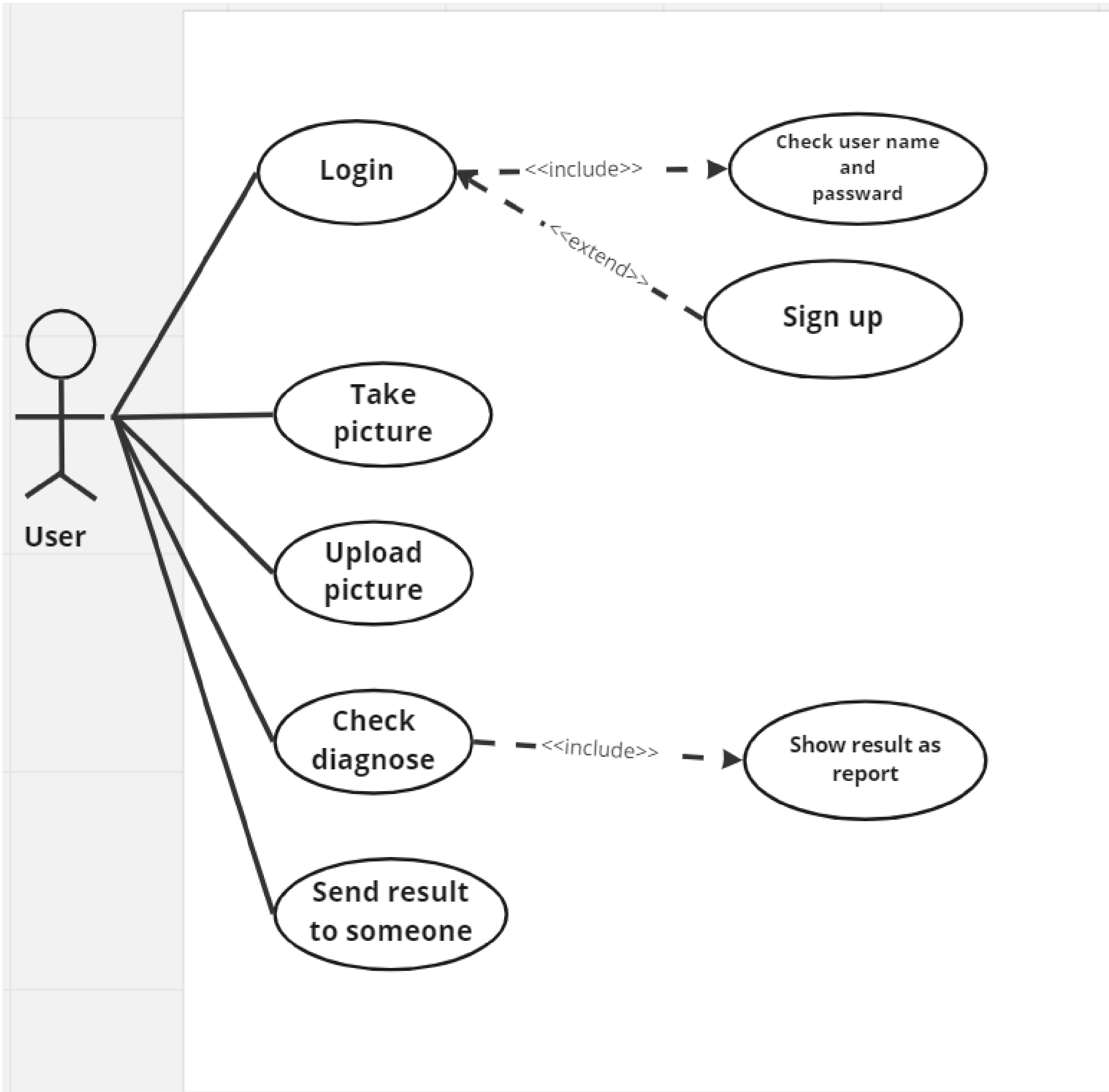


Figure.12: Use case diagram

- The main actor in the system is 'user'
- Actor 'user' has ability to log in in application, take or upload his/her MRI image and can find the result of diagnose as a report.

Class Diagram

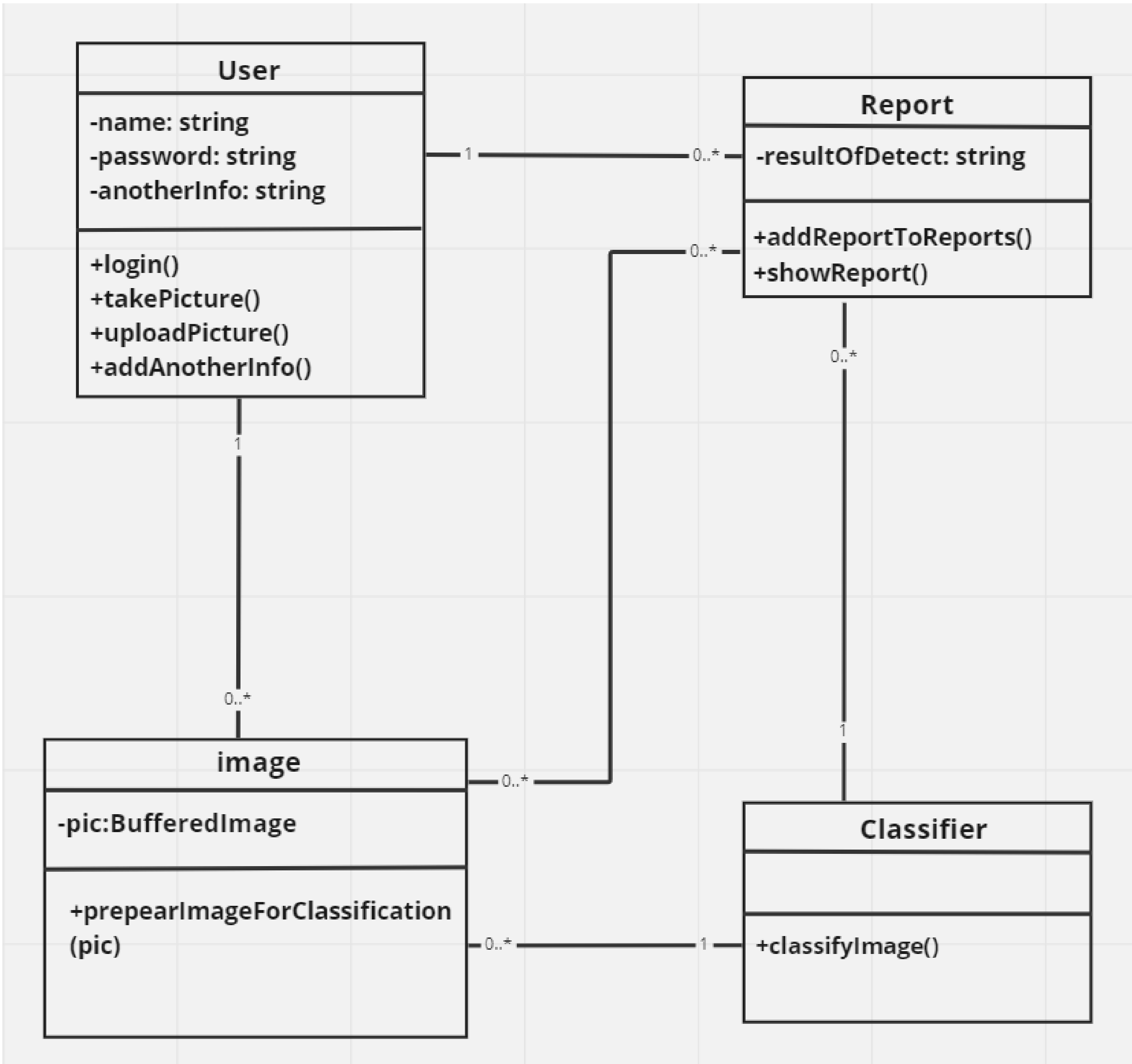


Figure.13: Class diagram

-Each user has many images can upload or take them and many images belong to one user.
-Each user has many reports and one report belongs to one user only.
-Each image has one classifier to detect and classification and classifier class can take many images.

Sequence Diagram

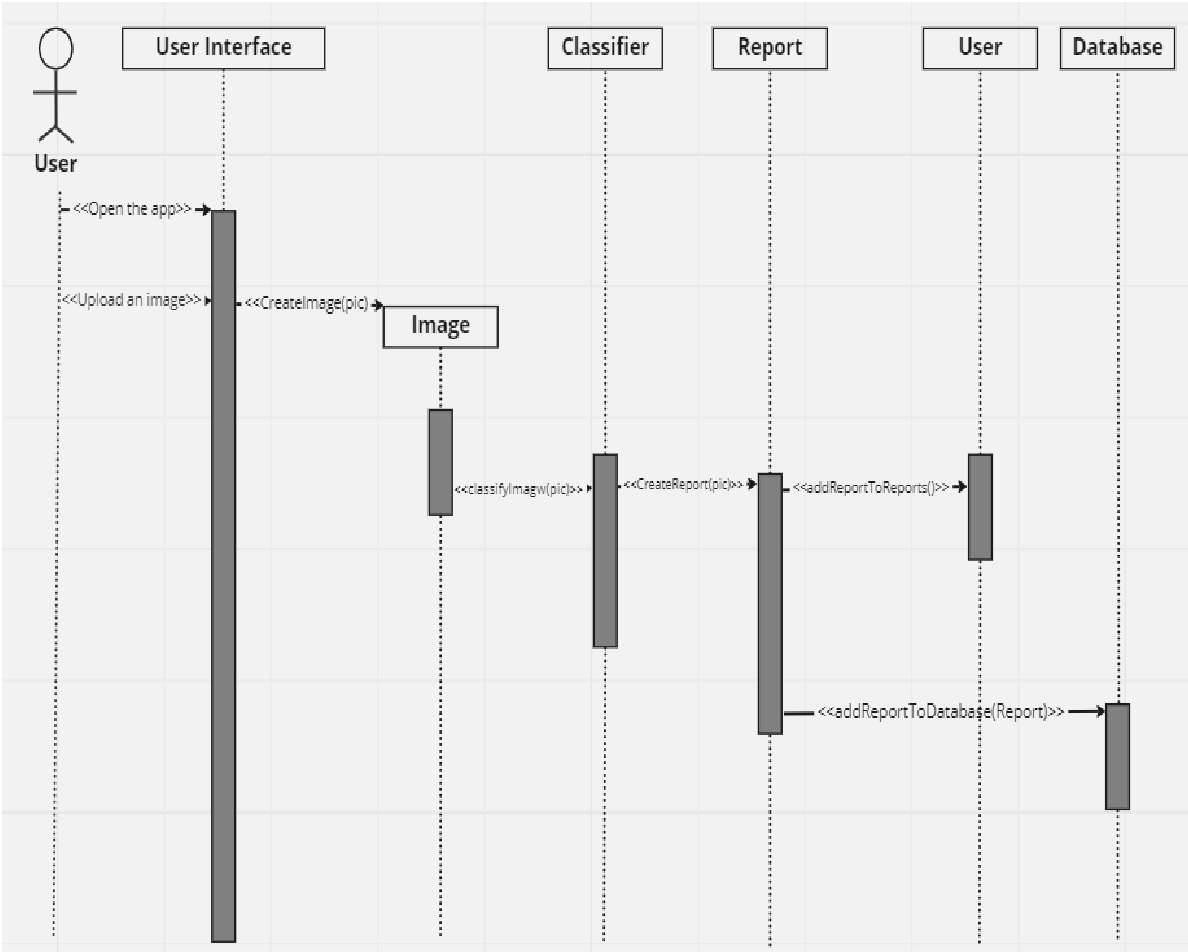


Figure.14: Sequence diagram

Discussion about the design

The provided diagrams show only the core features of the application and it is initial and can be modified later, all the diagrams are subject to modification and change in later cycles of the development.

The diagrams are simple because they are initial only 8 use cases with the core features and the classes those features need and a sequence diagram to present the lifeline of the classes.

Code of ethics

- Respecting data privacy and refraining from disclosing identifying information about the doctor or application's creators.
- Not to use technology to irritate, hurt, steal from, or violate the sanctities of others, or to attack their social and personal freedoms or to publicise or threaten them with information about their illnesses.
- It's crucial to consider what will happen to your gear, software, or data when you leave them while being aware of the risks posed by technology and abiding by ethical standards.

Chapter 4

Design, Implementation, and testing

Classification System

What is the classification system?

Classification system is the way to organize and categorize data by grouping similar items together and classifying data based on the shared information between the same category.

In this project:

Classification system is used to classify and recognize images that includes MS lesions and normal Images and put labels for each class, 1 for MS lesions and 0 for normal Images.

Current life cycle

In general, the life cycle of a species refers to the changes that occur from one developmental stage to the next as they pass from one generation to the next.
In software the life cycle of it called software development life cycle (SDLC)
Stages: (Planning - Requirements - Design – Implementation – Testing – Deployment – Maintenance)

-Planning: in this stage clarify the scope, goals, and objectives of the project. The project plan is created, and the feasibility of the project is assessed.

-Requirements: gathering and understand the requirements of software product and this information can be taken from stakeholders (users - Customers) and understand what the stakeholders need?

-Design: This stage involves the analyzing the feasibility of the project, identifying the constraints and developing a high-level design of the software. The outcome of this stage is a software design document that outlines the features, functionality, and architecture of the software.

-Implementation: in this stage the code is written, tested, and debugged until the software meets the requirements specified in the design document and the software is developed based on the design document created in the previous stage.

-Testing: the software is tested for bugs and errors to ensure it meets the requirements specified in the design document. The testing process includes unit testing, integration testing, system testing, and acceptance testing.

-Deployment: the software is deployed to the production environment, and the users start using it. The deployment process includes installation, configuration, and testing of the software in the production environment.

-Maintenance: In the last stage, the software is maintained and updated to meet the changing needs of the users. Maintenance includes bug fixing, performance optimization, feature enhancements, and updates to keep the software up-to-date and secure.

SDLC includes some techniques to manage these stages in the project and from these techniques Waterfall and agile.

Waterfall:

The Waterfall approach is a linear sequential software development model in which the entire project is divided into distinct phases, and each phase must be completed before the next one can begin.

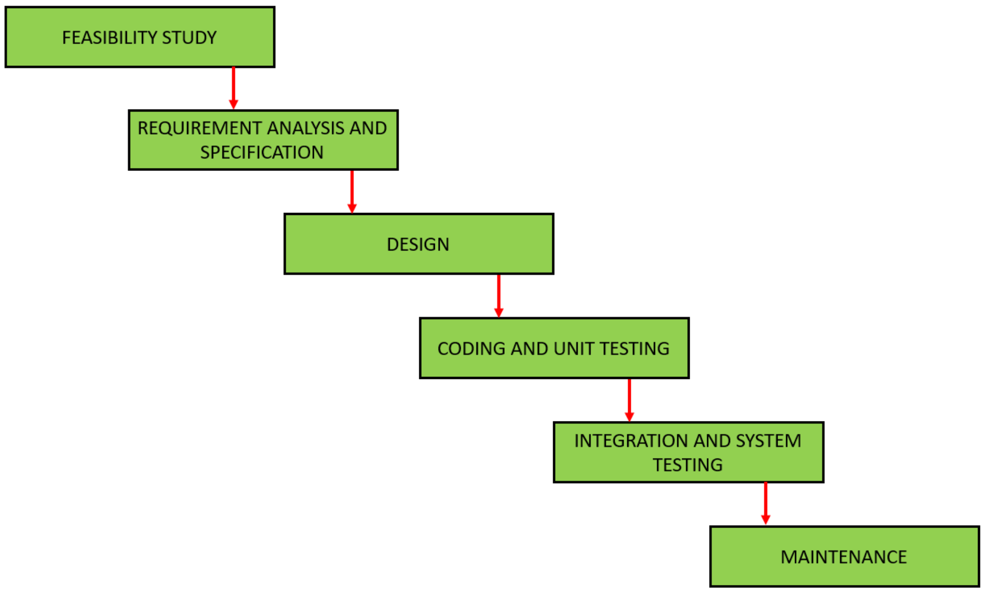


Figure15: phases of Waterfall model (Pal, n.d)

Agile:

Agile is a project management methodology that emphasizes collaboration, flexibility, and iterative development to deliver high-quality products or services that meet the customer's needs.

Why choosing agile approach:

Because the agile projects are divided into small, manageable chunks called sprints or iterations. The development team works on one iteration at a time, delivering working software at the end of each iteration.

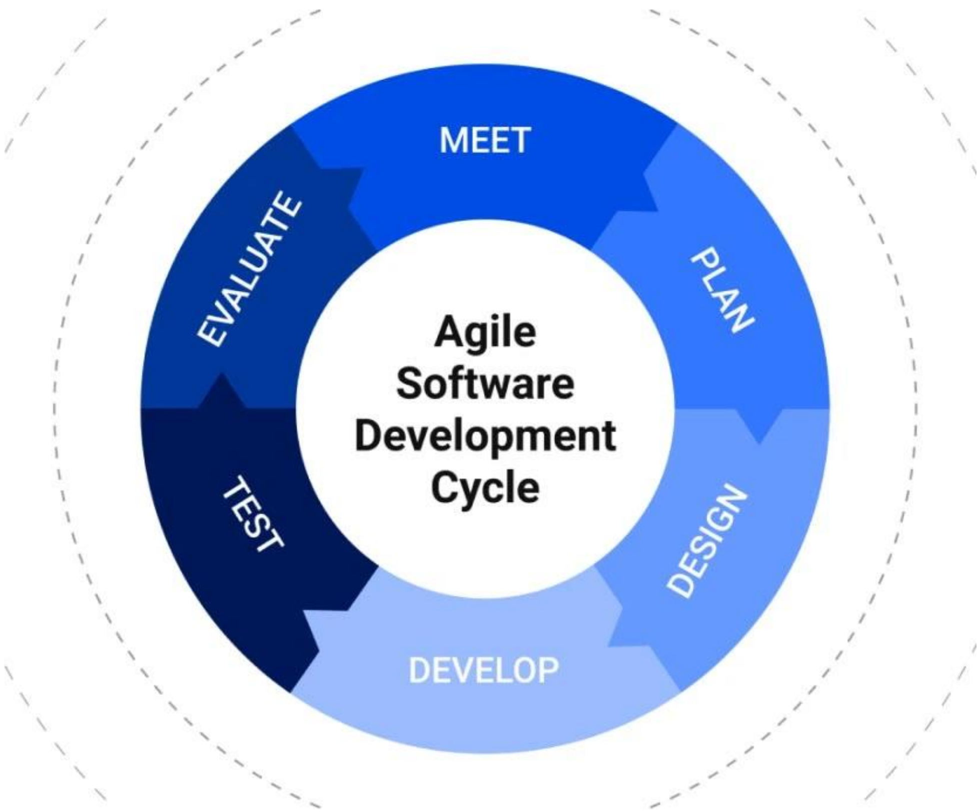


Figure16: phases of Agile model ***(Franciosi, 2020)***

Problem Statement

Multiple sclerosis (MS) is one of the most dangerous diseases that affects humans in our current era. Despite the fact that scientists and doctors do not know the main cause of the disease, they have discovered methods for diagnosing it. They have found that a malfunction in the immune system leads to the abnormal growth of white blood cells and their accumulation in any area of the brain, causing the erosion of nerve cells and the myelin substance responsible for wrapping and protecting these cells. Destroying these cells leads to a disruption in the vital functions of the affected person. The longer the treatment for this disease is delayed, the more the patient's condition deteriorates, making treatment more difficult.

Although this disease is very dangerous, it is difficult for many to detect it manually when diagnosing it using magnetic resonance imaging (MRI). Therefore, we have created a system that helps detect this disease by simply inputting the MRI image of the patient's brain. The system will process the image entered by the patient using Convolutional neural network (CNN), this is the technique from deep learning to analyze it and output the diagnosis automatically, indicating whether the person has MS or not.

Design Technique

Scale-Invariant Feature Transform (SIFT)

It is computer vision algorithm used to detect and extract features of images, SIFT working in four steps (tabmir, 2023):

Step one: Scale-space extrema detection: scale the representation of images in different scales to recognize the keypoints in all different scales.

Step two: Keypoint regularization: detect and determine keypoints are good to take for feature extraction.

Step three: Orientation assignment: descriptor is generated for each keypoint by computing histograms of gradient directions in a local neighborhood around the keypoint.

Step four: Descriptor generation: compare the feature vectors for keypoints in different images to find matches of keypoints between them.

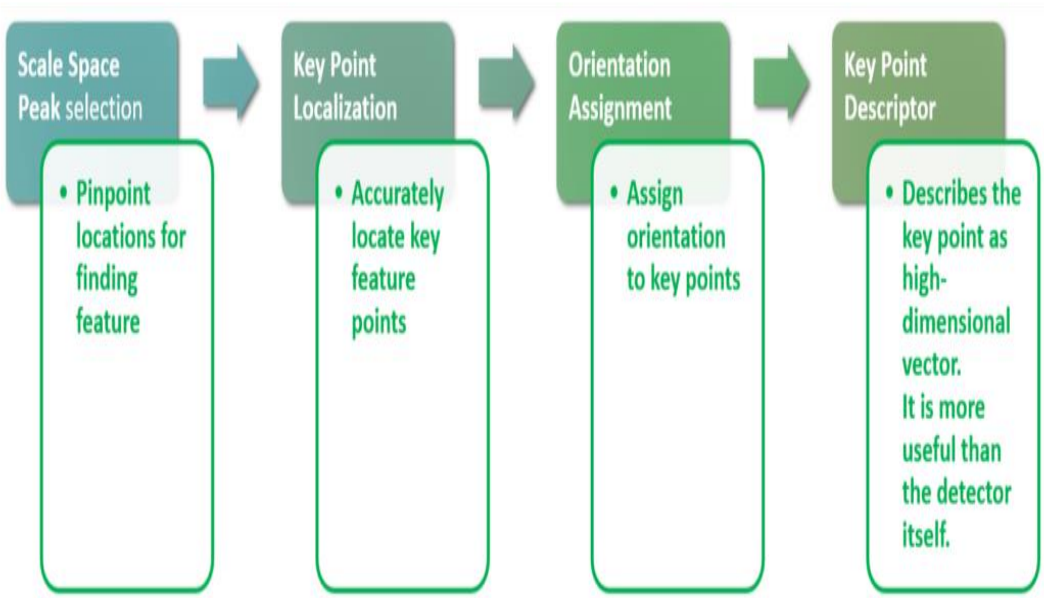


Figure 17: sequence of SIFT algorithm implementation.

After applying this step and run algorithm, the result is a set of feature vectors that can use it to be the input of any classification algorithm from machine learning techniques such as support vector machine (SVM) to classify the feature and determine if this image is MS lesions image or normal image.

pros:

1- The sift algorithm able to detect and match features in images at different scales and when rotating or skewing.

2- It uses a unique set of descriptors to represent features in the image, making them highly distinctive for accurate matching even if the image includes noise or occlusion.

Cons:

1- Not efficient with big data sets.
2- It requires setting several parameters such as size of Gaussian filter and the feature selection threshold. These parameters affect the result and need to tune it very well.
3- Intellectual property issues and the patent could be limited,it used in commercial or proprietary applications without having a license or permission from the patent holders.

Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) are a type of deep neural network that are used for image recognition and classification, it has the ability to learn automatically and extract the important feature from row image data without needing manual feature engineering.

From the structure and function of the human visual cortex, which processes visual information by analyzing small, overlapping regions of an image that inspired the idea of CNN.

CNN architecture consists of multiple layers that are three main layers (convolutional layer - pooling layer - flatten layer):

- convolutional layer: applied a set of filters or (called kernels) in image, each filter learns to detect a specific type of feature such as corners or edges and so on, in this filter also determine the size and the stride of filter to control the size of the output of the feature maps.
- pooling layer: in this layer helps to reduce the size of the feature maps and extract the most important information, it downsamples the output feature maps.
- fully connected layer: It takes the output of (convolutional layer and pooling layer) and transforms them into a vector of class scores, allowing the network to learn complex combinations of features (Gurucharan, 2022).

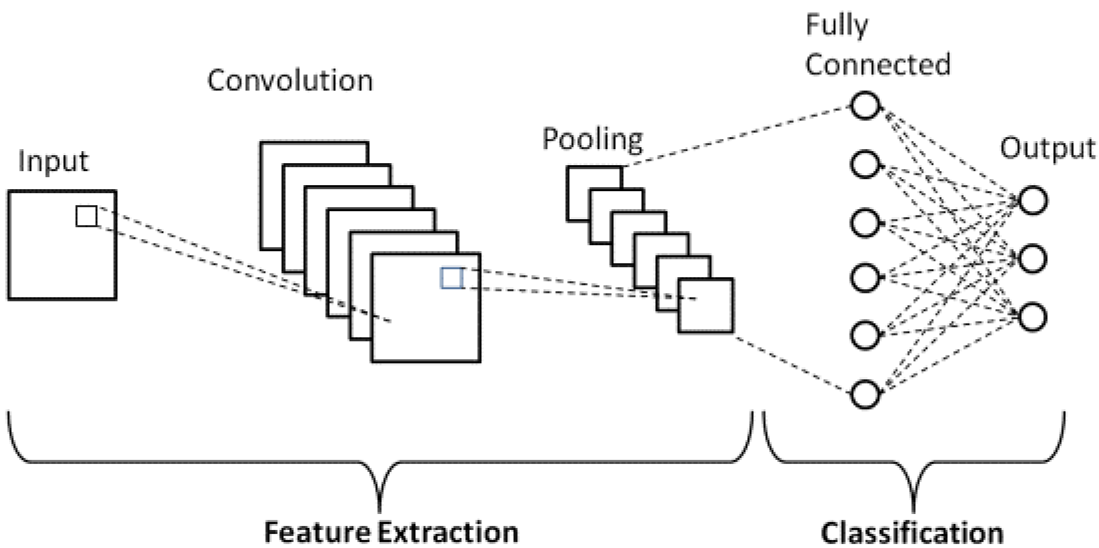


Figure 18: CNN architecture

Pros:

1. It is very good when training the big dataset.
2. CNNs can learn to recognize patterns and features that are difficult or impossible for humans to recognize.
3. Translation Invariance: CNNs can recognize the same feature regardless of where it appears in an image.
4. it can recognize the same feature regardless of where it appears in an image, thanks to their ability to perform convolution operations.
5. It achieved state-of-the-art performance on a wide range of image recognition and classification tasks, including object detection, facial recognition, and medical image analysis.

Cons:

1. CNNs require large amounts of training data to achieve high accuracy, and the quality of the data is crucial for successful training.
2. CNNs are often described as "black box" models because it can be difficult to understand how they arrive at their predictions. This lack of interpretability can be a challenge in applications where it is important to understand the reasoning behind the model's decisions.

Convolutional Neural Networks (CNN) with splitting data into train and test:

This is the common approach that the available data is randomly split into two sets: a training set and a testing set. The model is trained on the training set, and its performance is evaluated on the testing set.

Pros:

1- It is fast in implementation, easy to understand and it can be trained more quickly, allowing for faster experimentation and iteration.

Cons:

1- Overfitting: If the model is overfitting the training data, this approach may not identify it since the model is only evaluated on the testing set.

Convolutional Neural Networks (CNN) with splitting data into train, test, validation and adding data augmentation:

Pros:

1- By using validation set to fine-tune the model's hyperparameters, as well as techniques like data augmentation to increase the variety of the training data, CNN can be better at generalizing to new unseen data than simpler models.

2- It can learn features that are robust to variations in the input images, such as changes in scale, rotation, and lighting conditions, and this by using data augmentation that increase the data by increase the size of a training dataset by creating modified versions of the original images through various transformations such as (rotation - translation - scaling - flipping – etc).

Cons:

1- Splitting data into training, validation, and test sets requires additional time and effort.

Comparison between all techniques

Implementation and result of SIFT:

The code to implement the SIFT algorithm (to find features and description of image) with support vector machine (SVM) to classify features of image and show result:

```
from google.colab.patches import cv2_imshow

#importing libaries
import numpy as np
import pandas as pd
import random as rd
import os
import cv2

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
from PIL import Image

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

from sklearn import svm

from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

## read and load images

# 0 - Normal
# 1 - MS lesions

data = [] #creating a list for images
paths_normal = [] #creating a list for paths for normal images
paths_ms = [] #creating a list for paths for MS images
labels = [] #creating a list to put our 0 or 1 labels


#staring with the images with no MS   0
#
for r, d, f in os.walk(r'/content/drive/MyDrive/project notes/ready
dataset 0'):
    for file in f:
        if '.jpg' in file:
            paths_normal.append(os.path.join(r, file))

for path in paths_normal:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        labels.append(0)



#working with the images that have MS  1
for r, d, f in os.walk(r'/content/drive/MyDrive/project notes/FinalD
ataset'):
```

```
    for file in f:
        if '.jpg' in file:
            paths_ms.append(os.path.join(r, file))

for path in paths_ms:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        labels.append(1)


data = np.array(data)
data.shape

labels = np.array(labels)
labels = labels.reshape(len(labels),1)

print('data shape is:', data.shape)
print('labels shape is:', labels.shape)
```

```
data shape is: (53, 128, 128, 3)
labels shape is: (53, 1)
```

Figure 19: SIFT: result of data shape and labels shape

```
## just for testing
#print('Data:', data)
print('Paths for normal images:', paths_normal)
print('Paths for MS images:', paths_ms)
print('Labels:', labels)
```

```
Paths for normal images: ['/content/drive/MyDrive/project notes/ready dataset 0/image.0009.jpg', '/content/drive/MyDrive/project notes/ready dataset 0/image.0012.jpg', '/content/drive/M
Paths for MS images: ['/content/drive/MyDrive/project notes/FinalDataset/2- P,1 image 0015 L .jpg', '/content/drive/MyDrive/project notes/FinalDataset/3- P.2 image 0013 H .jpg', '/conten
'/content/drive/MyDrive/project notes/FinalDataset/32- p.30 image.0011(3) L .jpg', '/content/drive/MyDrive/project notes/FinalDataset/31- p.30 image.0011(2) L .jpg', '/content/drive/MyDr
Labels: [[0][0][0][0][0][0][0][0][0][0][0][0][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1][1]]
```

Figure 20: SIFT: result of paths_normal, paths_ms and labels

```
## apply SIFT algorithm
## in MS file
```

```
# Loop over each file path in the data list
for i, file_path in enumerate(paths_ms):
    #1 #2
    # Load the image in grayscale
    img = cv2.imread(file_path, 0)

## to split data with extension to (High, Low, Medium)
    if file_path.endswith('L .jpg'):
            #3
      # Apply a threshold to create a binary mask
      ret, thresh = cv2.threshold(img, 105, 255, cv2.THRESH_BINARY)

      #4
      # Find contours of white regions in the binary image
      contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

      #5
      # Create a mask of the same size as the image
      mask = np.zeros(img.shape, np.uint8)

      #6
      # Iterate over the contours and draw them on the mask
      for contour in contours:
          area = cv2.contourArea(contour)
          if area < 100:  # Example area threshold
              cv2.drawContours(mask, [contour], 0, 255, -1)

      #7
      # Apply the mask to the grayscale image to extract the region
of interest
      roi = cv2.bitwise_and(img, img, mask=mask)

      #8
      # Detect SIFT features in the region of interest
      sift = cv2.SIFT_create()
      kp, des = sift.detectAndCompute(roi, None)

      #9
      # Draw the keypoints on the original image
      img_with_keypoints = cv2.drawKeypoints(img, kp, None)
```

```
        # Display the original and processed images side by side using
 Matplotlib
        fig = plt.figure(figsize=(20, 10))
        plt.subplot(1, 3, 1)
        plt.imshow(img, cmap='gray')
        plt.title('Original Image')
        plt.subplot(1, 3, 2)
        plt.imshow(mask, cmap='gray')
        plt.title('Binary Mask')
        plt.subplot(1, 3, 3)
        plt.imshow(img_with_keypoints, cmap='gray')
        plt.title('img_with_keypoints')

        plt.show()
        # to check the path and name of file
        print(file_path)

    elif file_path.endswith('H .jpg'):
            #3
        # Apply a threshold to create a binary mask
        ret, thresh = cv2.threshold(img, 225, 255, cv2.THRESH_BINARY)

        #4
        # Find contours of white regions in the binary image
        contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        #5
        # Create a mask of the same size as the image
        mask = np.zeros(img.shape, np.uint8)

        #6
        # Iterate over the contours and draw them on the mask
        for contour in contours:
            area = cv2.contourArea(contour)
            if area < 100:
                cv2.drawContours(mask, [contour], 0, 255, -1)

        #7
        # Apply the mask to the grayscale image to extract the region
of interest
        roi = cv2.bitwise_and(img, img, mask=mask)

        #8
        # Detect SIFT features in the region of interest
```

```
        sift = cv2.SIFT_create()
        kp, des = sift.detectAndCompute(roi, None)

        #9
        # Draw the keypoints on the original image
        img_with_keypoints = cv2.drawKeypoints(img, kp, None)



        # Display the original and processed images side by side using
   Matplotlib
        fig = plt.figure(figsize=(20, 10))
        plt.subplot(1, 3, 1)
        plt.imshow(img, cmap='gray')
        plt.title('Original Image')
        plt.subplot(1, 3, 2)
        plt.imshow(mask, cmap='gray')
        plt.title('Binary Mask')
        plt.subplot(1, 3, 3)
        plt.imshow(img_with_keypoints, cmap='gray')
        plt.title('img_with_keypoints')

        plt.show()
        print(file_path)
      elif file_path.endswith('M .jpg'):
                     #3
        # Apply a threshold to create a binary mask
        ret, thresh = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY)

        #4
        # Find contours of white regions in the binary image
        ## to detect spcific features in image by draw it to put the c
  olor wight
        contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
  cv2.CHAIN_APPROX_SIMPLE)

        #5
        # Create a mask of the same size as the image
        ## unit8 --> to put imge in 8 unsigned integer bits
        ## it makes performance good and decrease the storage in memeo
  ry
        mask = np.zeros(img.shape, np.uint8)

        #6
        # Iterate over the contours and draw them on the mask
```

```
    for contour in contours:
        area = cv2.contourArea(contour)
        if area < 100:
            ## apply maske,  array of contour and in region betwee
n 0-255
            ## -1 --> draw the contour in white color
            cv2.drawContours(mask, [contour], 0, 255, -1)

    #7
    # Apply the mask to the grayscale image to extract the region
of interest
    roi = cv2.bitwise_and(img, img, mask=mask)

    #8
    # Detect SIFT features in the region of interest
    sift = cv2.SIFT_create()
    ## none --> mo more calculation needs to compute des and kp
    kp, des = sift.detectAndCompute(roi, None)

    #9
    # Draw the keypoints on the original image
    ## none --
> output of new image will have the same size and type
    img_with_keypoints = cv2.drawKeypoints(img, kp, None)



    # Display the original and processed images side by side using
 Matplotlib
    fig = plt.figure(figsize=(20, 10))
    plt.subplot(1, 3, 1)
    plt.imshow(img, cmap='gray')
    plt.title('Original Image')
    plt.subplot(1, 3, 2)
    plt.imshow(mask, cmap='gray')
    plt.title('Binary Mask')
    plt.subplot(1, 3, 3)
    plt.imshow(img_with_keypoints, cmap='gray')
    plt.title('img_with_keypoints')

    plt.show()
    print(file_path)
```

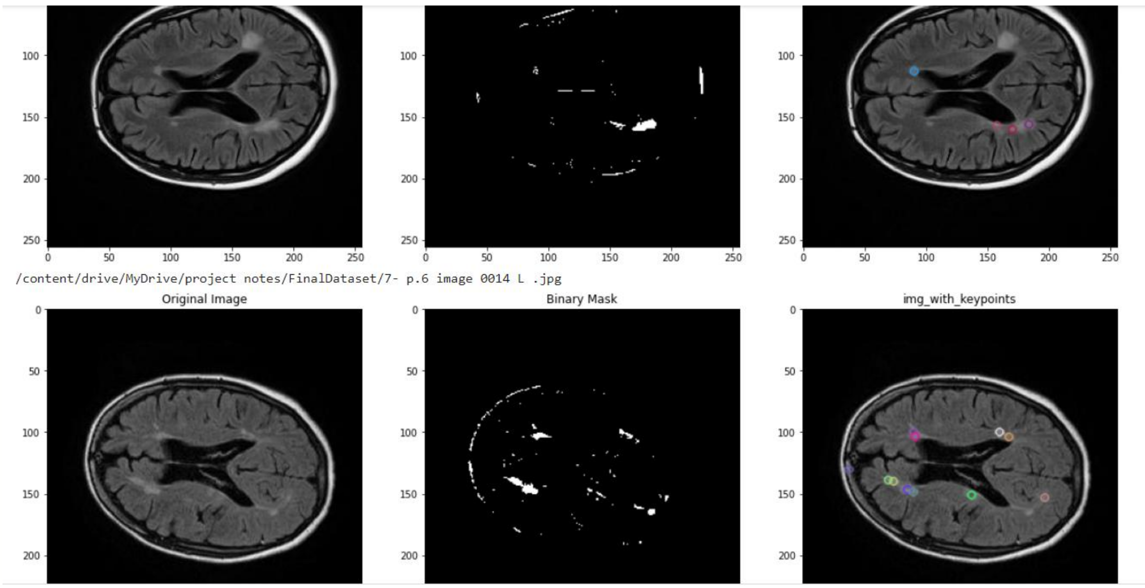


Figure 21: Result of SIFT in MS images

```
print(des)
```

```
[[  0.   0.   1. ... 114.   3.   0.]
 [  1.  13.  15. ...  43.  45.   2.]
 [  3.   5.  21. ...   5.   0.   2.]
 ...
 [  2.   0.   0. ...   0.   0.   0.]
 [  2.   0.   0. ...   0.   0.   0.]
 [  1.   4.   5. ...   0.   1.   5.]]
```

Figure 22: features of MS images

```
## apply SIFT algorithm
## in normal file

# Loop over each file path in the data list
for i, file_path in enumerate(paths_normal):
    #1 #2
    # Load the image in grayscale
    img = cv2.imread(file_path, 0)

    #3
    # Apply a threshold to create a binary mask
    ret_norm, thresh_norm = cv2.threshold(img, 50, 255, cv2.THRESH_B
INARY)
```

```
    #4
    # Find contours of white regions in the binary image
    contours_norm, hierarchy_norm = cv2.findContours(thresh_norm, cv
2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    #5
    # Create a mask of the same size as the image
    mask = np.zeros(img.shape, np.uint8)

    #6
    # Iterate over the contours and draw them on the mask
    for contour_norm in contours_norm:
        area = cv2.contourArea(contour_norm)
        if area < 100:  # Example area threshold
            cv2.drawContours(mask, [contour_norm], 0, 255, -1)

    #7
    # Apply the mask to the grayscale image to extract the region of
 interest
    roi_norm = cv2.bitwise_and(img, img, mask = mask)

    #8
    # Detect SIFT features in the region of interest
    sift_norm = cv2.SIFT_create()
    kp_norm, des_norm = sift_norm.detectAndCompute(roi_norm, None)

    #9
    # Draw the keypoints on the original image
    img_with_keypoints_norm = cv2.drawKeypoints(img, kp_norm, None)



    # Display the original and processed images side by side using M
atplotlib
    fig = plt.figure(figsize=(20, 10))
    plt.subplot(1, 3, 1)
    plt.imshow(img, cmap='gray')
    plt.title('Original Image')
    plt.subplot(1, 3, 2)
    plt.imshow(mask, cmap='gray')
    plt.title('Binary Mask')
    plt.subplot(1, 3, 3)
    plt.imshow(img_with_keypoints_norm, cmap='gray')
    plt.title('img_with_keypoints')
```

```
print(file_path)

plt.show()
```

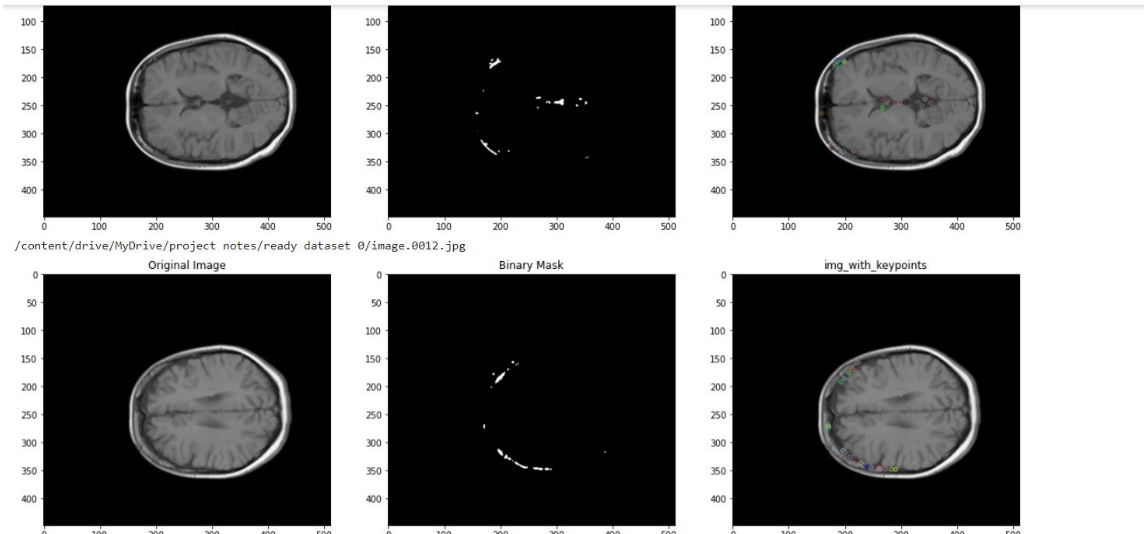


Figure 23: Result of SIFT in normal image

```
print(des_norm)
```

```
[[ 2.  0.  0. ...  1.  0.  0.]
 [ 0.  0.  0. ...  2.  0.  0.]
 [12.  0.  0. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ...  0.  0.  0.]
 [13.  0.  0. ...  0.  0.  0.]
 [10.  0.  0. ...  0.  0.  0.]]
```

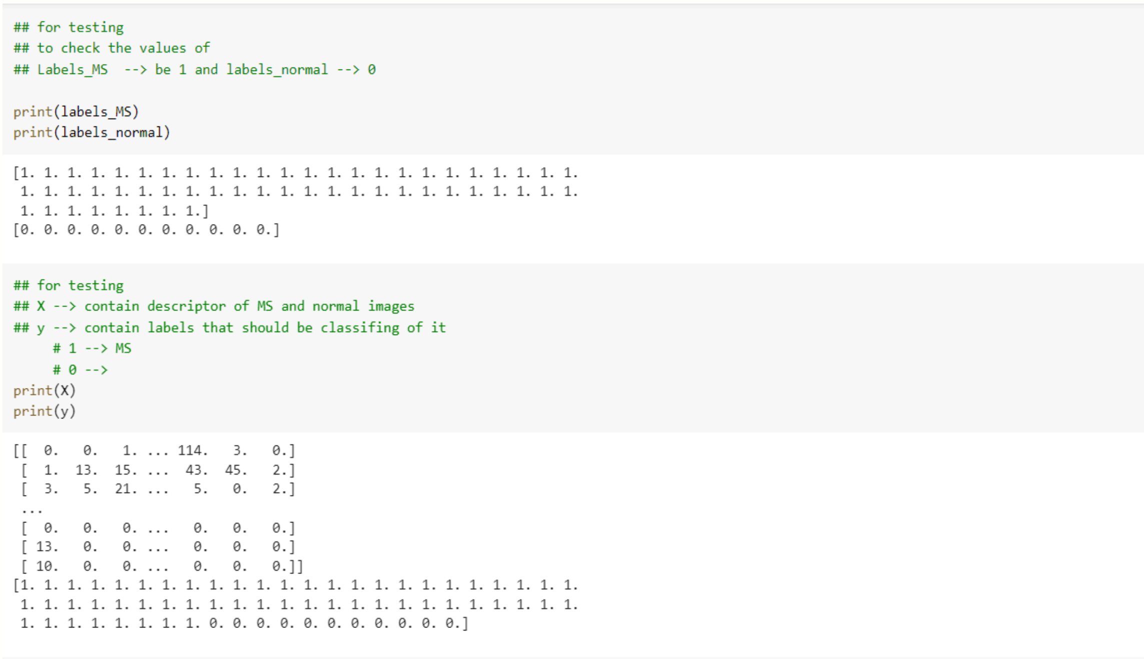
Figure 24: Features in normal image

```
## for testing
## to check the values of
## Labels_MS  --> be 1 and labels_normal --> 0

print(labels_MS)
print(labels_normal)


## for testing
## X --> contain descriptor of MS and normal images
## y --> contain labels that should be classifing of it
```

```
     # 1 --> MS
     # 0 -->
print(X)
print(y)
```



```
## for testing
## to check the values of
## Labels_MS  --> be 1 and labels_normal --> 0

print(labels_MS)
print(labels_normal)
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
## for testing
## X --> contain descriptor of MS and normal images
## y --> contain labels that should be classifing of it
     # 1 --> MS
     # 0 -->
print(X)
print(y)
```

```
[[  0.   0.   1. ... 114.   3.   0.]
 [  1.  13.  15. ...  43.  45.   2.]
 [  3.   5.  21. ...   5.   0.   2.]
 ...
 [  0.   0.   0. ...   0.   0.   0.]
 [ 13.   0.   0. ...   0.   0.   0.]
 [ 10.   0.   0. ...   0.   0.   0.]]
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Figure 25: Result of labels

```
## 2
# split the data into training, validation, and test sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.
2, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_val, y_val, test_s
ize=0.5, random_state=42)

#3
## check the best value of C, to make the model more accurate
# define a range of values for the C hyperparameter
C_values = [0.1, 1, 10, 100]

best_score = 0
best_C = 0

# evaluate the performance of the SVM model for different values of
C on the validation set
for C in C_values:
    model = SVC(kernel='rbf', C=C, gamma='scale')
    model.fit(X_train, y_train)
    y_pred = model.predict(X_val)
```

```
    score = accuracy_score(y_val, y_pred)
    if score > best_score:
        best_score = score
        best_C = C

# train a final SVM model using the best value of C on the full trai
ning set
final_model = SVC(kernel='rbf', C=best_C, gamma='scale')
final_model.fit(X_train, y_train)
y_pred = final_model.predict(X_test)
test_score = accuracy_score(y_test, y_pred)

#4
print("Best C value: ", best_C)
print("Validation accuracy: ", best_score)
print("Accuracy: ", test_score)
```

And the result of this code:

```
#4
print("Best C value: ", best_C)
print("Validation accuracy: ", best_score)
print("Accuracy: ", test_score)

Best C value:  100
Validation accuracy:  1.0
Accuracy:  0.7142857142857143
```

Figure 26: The result of SIFT algorithm with SVM

but this result not good because the data is very small (in this code train in the 53 images only and when need to increase the data SIFT cannot implement will and give the wrong accuracy)
and the second problem, SIFT algorithm is sensitive to changes in lighting conditions and image noise, which can affect the accuracy of the key point detection and feature extraction, And this appeared when the data was divided into three sections(High (H) - Low (L) - (Medium (M))) according to the intensity of each image's lighting through manual experimentation and measuring each of the threshold for each image, and this is not practical and did not solve the problem automatically.

Implementation and result of CNN with splitting data into training and testing sets:

After importing and preprocessing data start to create and train CNN model, this section focuses on the splitting part of data and the result of training, and the next section is implementation section we can know more about the previous steps before implement splitting data and how create CNN model.

Splitting data step:

▾ Splitting data

```
#splitting data


x_train,x_test,y_train,y_test = train_test_split(data, labels, test_size=0.3, shuffle=True, random_state=7)

print("shape of our training data:",x_train.shape)
print("shape of our training labels:",y_train.shape)
print("shape of our test data:",x_test.shape)
print("shape of our test labels:",y_test.shape)
```

```
shape of our training data: (97, 128, 128, 3)
shape of our training labels: (97, 1)
shape of our test data: (42, 128, 128, 3)
shape of our test labels: (42, 1)
```

Figure 27: Splitting data in CNN with splitting data into training and testing sets

Training the data in CNN model

```
Epoch 17/150
4/4 - 5s - loss: 0.6642 - accuracy: 0.5876 - val_loss: 0.5977 - val_accuracy: 0.7857 - 5s/epoch - 1s/step
Epoch 18/150
4/4 - 12s - loss: 0.6588 - accuracy: 0.5876 - val_loss: 0.6071 - val_accuracy: 0.7143 - 12s/epoch - 3s/step
Epoch 19/150
4/4 - 5s - loss: 0.6572 - accuracy: 0.6598 - val_loss: 0.6045 - val_accuracy: 0.7143 - 5s/epoch - 1s/step
Epoch 20/150
4/4 - 7s - loss: 0.6491 - accuracy: 0.6701 - val_loss: 0.6060 - val_accuracy: 0.7143 - 7s/epoch - 2s/step
Epoch 21/150
4/4 - 7s - loss: 0.6429 - accuracy: 0.7113 - val_loss: 0.5763 - val_accuracy: 0.7857 - 7s/epoch - 2s/step
Epoch 22/150
4/4 - 5s - loss: 0.6324 - accuracy: 0.6495 - val_loss: 0.5589 - val_accuracy: 0.7857 - 5s/epoch - 1s/step
Epoch 23/150
4/4 - 9s - loss: 0.6445 - accuracy: 0.6392 - val_loss: 0.5538 - val_accuracy: 0.7857 - 9s/epoch - 2s/step
Epoch 24/150
4/4 - 5s - loss: 0.6273 - accuracy: 0.6495 - val_loss: 0.5830 - val_accuracy: 0.7143 - 5s/epoch - 1s/step
Epoch 25/150
4/4 - 6s - loss: 0.6452 - accuracy: 0.6598 - val_loss: 0.6775 - val_accuracy: 0.4524 - 6s/epoch - 2s/step
Epoch 26/150
4/4 - 7s - loss: 0.6975 - accuracy: 0.5773 - val_loss: 0.6966 - val_accuracy: 0.4524 - 7s/epoch - 2s/step
Epoch 27/150
4/4 - 5s - loss: 0.6711 - accuracy: 0.5773 - val_loss: 0.5728 - val_accuracy: 0.7619 - 5s/epoch - 1s/step
Epoch 28/150
4/4 - 8s - loss: 0.6322 - accuracy: 0.6289 - val_loss: 0.5638 - val_accuracy: 0.7143 - 8s/epoch - 2s/step
Epoch 29/150
4/4 - 5s - loss: 0.6702 - accuracy: 0.5258 - val_loss: 0.5749 - val_accuracy: 0.6667 - 5s/epoch - 1s/step
Epoch 30/150
4/4 - 5s - loss: 0.6948 - accuracy: 0.5258 - val_loss: 0.5816 - val_accuracy: 0.6429 - 5s/epoch - 1s/step
Epoch 31/150
4/4 - 8s - loss: 0.6972 - accuracy: 0.5258 - val_loss: 0.5741 - val_accuracy: 0.6667 - 8s/epoch - 2s/step
Epoch 32/150
4/4 - 5s - loss: 0.6770 - accuracy: 0.5464 - val_loss: 0.5875 - val_accuracy: 0.7619 - 5s/epoch - 1s/step
Epoch 33/150
4/4 - 8s - loss: 0.6485 - accuracy: 0.6186 - val_loss: 0.6156 - val_accuracy: 0.7857 - 8s/epoch - 2s/step
```

Figure 28: CNN training model after splitting data into train and test set

Result

Result

```
[ ] # Print accuracy
    print("Training accuracy:", history.history['accuracy'][-1])
    print("Validation accuracy:", history.history['val_accuracy'][-1])

    test_accuracy = history.history['val_accuracy'][-1]
    print("Test accuracy:", test_accuracy)
```

```
Training accuracy: 0.6185566782951355
Validation accuracy: 0.7857142686843872
Test accuracy: 0.7857142686843872
```

Figure 29: Result of CNN training model after splitting data into train and test set

In this result we we can see the big difference between the training and test accuracy and this is refer to the overfitting in training model.

The summary of comparison

	Sift algorithm	**CNN model with splitting data into test and training sets**	**CNN model with splitting data into test, validation and training sets**
How can detect features in images?	Manually	automatically	automatically
Sensitive of noise?	Yes	No	No
Size of data needs to train:	Small data In this experiment (data = 53 images)	Large data	Large data
Result:	Test accuracy=71%	Test accuracy = 78 % Training accuracy = 61%	Test accuracy = 79 % Training accuracy = 77%

Table 3: Summary of comparison between 3 techniques

The code and implementation of CNN model with splitting data into test, validation and training sets in the next section.

Implementation

Model Code:

I using google colab to implement The code

- Install opencv python library, to use it in visualizing the data

```
!pip install opencv-python
!pip install --upgrade opencv-python
```

Output:

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.7.0.72)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.22.4)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.7.0.72)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.22.4)
```

Figure 30: install Libraries

-Show the version of it

```
import cv2
print(cv2.__version__)
```

Output:

```
4.7.0
```

Figure 31: CV2 library version

- connect the notebook with drive to load the data in the next steps

```
from google.colab import drive
drive.mount('/content/drive')
```

-Importing the necessary libraries

```
 #importing libaries

#Numpy and Pandas are libraries used for data manipulation and
analysis
import numpy as np
import pandas as pd
import random as rd
import os

#Matplotlib and Seaborn are used for data visualization.
```

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

#PIL (Python Imaging Library) is used for image processing tasks
from PIL import Image

#For splitting data and evaluating model performance


from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

#For building and training deep learning models
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
from keras.metrics import accuracy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import regularizers

#setting seed for reproducibility
from numpy.random import seed
seed(25)
tf.random.set_seed(50)
```

-Loading data

```
# 0 - Normal
# 1 - MS lesions

data = [] #creating a list for images
paths = [] #creating a list for paths
labels = [] #creating a list to put our 0 or 1 labels

#staring with the images that have MS lesions
for r, d, f in os.walk(r'/content/drive/MyDrive/project
notes/Dataset_Diagnose_MS/Yes_MS_lesions'):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
```

```
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        labels.append(1)

#now working with the images with no MS lesions
paths = []
for r, d, f in os.walk(r'/content/drive/MyDrive/project
notes/Dataset_Diagnose_MS/No_MS_lesions'):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    img = img.resize((128,128))
    img = np.array(img)
    if(img.shape == (128,128,3)):
        data.append(np.array(img))
        labels.append(0)

data = np.array(data)
data.shape

labels = np.array(labels)
labels = labels.reshape(305,1)

print('data shape is:', data.shape)
print('labels shape is:', labels.shape)
```

output:

```
data shape is: (305, 128, 128, 3)
labels shape is: (305, 1)
```

Figure 32: output of data and labels shape

Data preprocessing:

```
##just for testing
## to show if values store correct or not
print("data list", data)
print("______________________________")
print("paths",paths)
print("______________________________")
print("labels",labels)
```

output:

```
data list [[[[  1   1   1]
   [  1   1   1]
   [  0   0   0]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

  [[  1   1   1]
   [  1   1   1]
   [  2   2   2]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

  [[  1   1   1]
   [  1   1   1]
   [  1   1   1]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

  ...

  [[  2   2   2]
   [  2   2   2]
   [  2   2   2]
   ...
   [  1   1   1]
   [  1   1   1]
   [  1   1   1]]

  [[  2   2   2]
   [  2   2   2]

   [  0   0   0]
   [  1   1   1]
   [  1   1   1]]

  [[  0   0   0]
   [  0   0   0]
   [  1   1   1]
   ...
   [  0   0   0]
   [  1   1   1]
   [  1   1   1]]]

 [[[  0   0   0]
   [  0   0   0]
   [  0   0   0]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

  [[  0   0   0]
   [  0   0   0]
   [  0   0   0]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

  [[  0   0   0]
   [  0   0   0]
   [  0   0   0]
   ...
   [  0   0   0]
   [  0   0   0]
   [  0   0   0]]

 [[[122 122 122]
   [ 23  23  23]
   [ 21  21  21]
   ...
   [ 12  12  12]
   [ 11  11  11]
   [ 12  12  12]]

  [[124 124 124]
   [ 53  53  53]
   [ 34  34  34]
   ...
   [ 11  11  11]
   [ 11  11  11]
   [ 12  12  12]]

  [[ 44  44  44]
   [ 87  87  87]
   [ 43  43  43]
   ...
   [ 11  11  11]
   [ 12  12  12]
   [ 12  12  12]]

  ...

  [[ 10  10  10]
   [ 12  12  12]
   [ 11  11  11]
   ...
   [ 11  11  11]
   [ 10  10  10]
   [ 11  11  11]]

  [[ 12  12  12]
   [ 12  12  12]
   [ 13  13  13]

   [ 63  67  68]
   [140 144 145]]

  [[  3   3   3]
   [  3   3   3]
   [  3   3   3]
   ...
   [ 11  15  12]
   [ 16  19  15]
   [ 41  44  39]]

  ...

  [[  2   2   2]
   [  2   2   2]
   [  2   2   2]
   ...
   [  3   3   3]
   [  3   3   3]
   [  2   2   2]]

  [[  2   2   2]
   [  2   2   2]
   [  2   2   2]
   ...
   [  2   2   2]
   [  2   2   2]
   [  2   2   2]]

  [[  2   2   2]
   [  2   2   2]
   [  2   2   2]
   ...
   [  1   1   1]
   [  1   1   1]
   [  1   1   1]]]]
```

Figure 33: output 1 for data preprocessing

```
Paths for MS images: ['/content/drive/MyDrive/project notes/FinalDataset/2- P,1 image 0015 L .jpg', '/content/drive/MyDrive/project notes/FinalDataset/3- P.2 image 0013 H .jpg', '/content/drive/MyDrive/pro
'/content/drive/MyDrive/project notes/FinalDataset/21- p.22 image.0015 L .jpg', '/content/drive/MyDrive/project notes/FinalDataset/23- p.27 image.0016(1) L .jpg', '/content/drive/MyDrive/project notes/Fina
 '/content/drive/MyDrive/project notes/FinalDataset/25- p.28 image 0015(1) H .jpg', '/content/drive/MyDrive/project notes/FinalDataset/30- p.30 image.0011(1) L .jpg',
'/content/drive/MyDrive/project notes/FinalDataset/32- p.30 image.0011(3) L .jpg', '/content/drive/MyDrive/project notes/FinalDataset/31- p.30 image.0011(2) L .jpg', '/content/drive/MyDrive/project notes/F
labels = [[1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1],
[1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1],/[1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1],
[1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1],
[1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0],
```

Figure 34: output 2 for data preprocessing

```python
#getting the max of the array
print(np.max(data))
#getting the min of the array
print(np.min(data))
```

output:

```
255
0
```

Figure 35: output 3 for data preprocessing

-Sca;ing image data between 0 and 1 to improve the performance of training in next steps

```
#reducing the data to between 1 and 0
data = data / 255.00
#getting the max of the array
print(np.max(data))
#getting the min of the array
print(np.min(data))
```

Output:

```
1.0
0.0
```

Figure 36: output 4 for data preprocessing

-vesualizing images

```
# show images
for i in range(5):
    fig = plt.figure(figsize=(20,20))
    plt.subplot(5,5,i+1)
    image = plt.imshow(data[i])
    plt.show(image)
```

output:

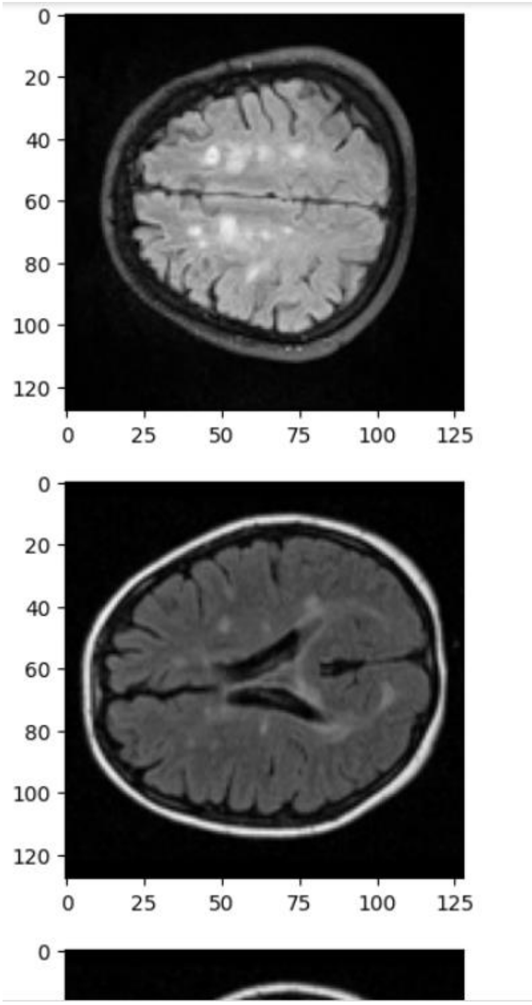


Figure 37: Data visualizing output

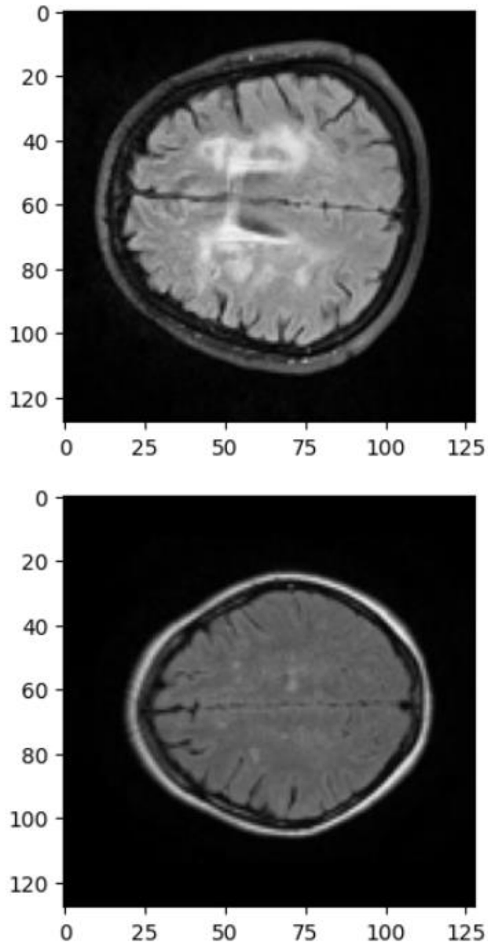


Figure 38: Data visualizing output 2

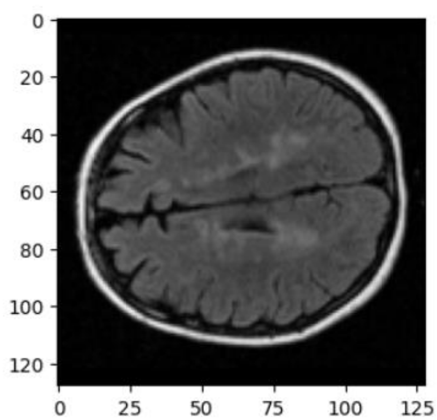


Figure 39: Data visualizing output 3

-Splitting data

```
# Splitting the data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(data, labels,
test_size=0.15, shuffle=True, random_state=7)

# Splitting the train data into train and validation sets
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.15, shuffle=True, random_state=7)
```

```
## testing 15%
## validation 15%
## training 70%

print("Shape of our training data:", x_train.shape)
print("Shape of our training labels:", y_train.shape)
print("Shape of our validation data:", x_val.shape)
print("Shape of our validation labels:", y_val.shape)
print("Shape of our test data:", x_test.shape)
print("Shape of our test labels:", y_test.shape)
```

Output:

```
Shape of our training data: (220, 128, 128, 3)
Shape of our training labels: (220, 1)
Shape of our validation data: (39, 128, 128, 3)
Shape of our validation labels: (39, 1)
Shape of our test data: (46, 128, 128, 3)
Shape of our test labels: (46, 1)
```

Figure 40: Data splitting output

-Adding data augmentation, to create new variations of the training data and can help to reduce overffiting

```
# adding data augmentation
## after importing library from tensorflow.keras.preprocessing.image
import ImageDataGenerator

# Define the data augmentation parameters
train_datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

```
# Fit the data augmentation parameters on the training data
train_datagen.fit(x_train)

# Create the augmented training data generator
train_generator = train_datagen.flow(
    x_train, y_train, batch_size=64, shuffle=True, seed=7)
# Create the validation data generator
val_datagen = ImageDataGenerator()
val_generator = val_datagen.flow(
    x_val, y_val, batch_size=64, shuffle=False)
```

-Create convolutional neural network (CNN) model

```
# cnn model

model = keras.Sequential([

    layers.Conv2D(
        filters=32,
        kernel_size=(5,5),
        activation="relu",
        padding='same',
        input_shape=[128, 128, 3],
        kernel_regularizer=regularizers.l2(0.001) # add L2
regularization
    ),
    layers.MaxPool2D(),

    layers.Conv2D(
        filters=64,
        kernel_size=(3,3),
        activation="relu",
        padding='same',
        kernel_regularizer=regularizers.l2(0.001) # add L2
regularization
    ),
    layers.MaxPool2D(),

    layers.Conv2D(
        filters=128,
        kernel_size=(3,3),
        activation="relu",
        padding='same',
```

```
        kernel_regularizer=regularizers.l2(0.001) # add L2 
regularization
    ),
    layers.MaxPool2D(),

    layers.Conv2D(
        filters=128,
        kernel_size=(3,3),
        activation="relu",
        padding='same',
        kernel_regularizer=regularizers.l2(0.001) # add L2 
regularization
    ),
    layers.MaxPool2D(),

    layers.Flatten(),
    layers.Dropout(0.8),
    layers.Dense(units=256, activation="relu"),
    layers.Dense(units=1, activation="sigmoid"),
])



# to show information of CNN
model.summary()
```

Output:

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 128, 128, 32)      2432

 max_pooling2d_12 (MaxPoolin  (None, 64, 64, 32)       0
 g2D)

 conv2d_13 (Conv2D)          (None, 64, 64, 64)        18496

 max_pooling2d_13 (MaxPoolin  (None, 32, 32, 64)       0
 g2D)

 conv2d_14 (Conv2D)          (None, 32, 32, 128)       73856

 max_pooling2d_14 (MaxPoolin  (None, 16, 16, 128)      0
 g2D)

 conv2d_15 (Conv2D)          (None, 16, 16, 128)       147584

 max_pooling2d_15 (MaxPoolin  (None, 8, 8, 128)        0
 g2D)

 flatten_3 (Flatten)         (None, 8192)              0

 dropout_3 (Dropout)         (None, 8192)              0

 dense_6 (Dense)             (None, 256)               2097408

 dense_7 (Dense)             (None, 1)                 257

=================================================================
Total params: 2,340,033
Trainable params: 2,340,033
Non-trainable params: 0
_________________________________________________________________
```

Figure41: Summary of CNN model

-configure and optimize the CNN model for training

```
# configure the model for training
model.compile(
    optimizer=tf.keras.optimizers.Adam(epsilon=0.01),
    loss='hinge',
    metrics=['accuracy']
)

#including early stopping to prevent overfitting
early_stopping = keras.callbacks.EarlyStopping(
    patience=10,
    min_delta=0.0001,
    restore_best_weights=True,
)
```

-Training the model with batch size = 32 and it means the model in each epoch take 32 random image and training on it and epochs =200 it means the number of iterations to train 200

```
history = model.fit(
    train_generator,
    batch_size=32,
    epochs=200,
    validation_data=val_generator,
    callbacks=[early_stopping],
    verbose=2
)
```

Output(Some screenshots from training result)

```
Epoch 1/200
4/4 - 10s - loss: 0.9945 - accuracy: 0.3909 - val_loss: 0.9664 - val_accuracy: 0.5897 - 10s/epoch - 3s/step
Epoch 2/200
4/4 - 8s - loss: 0.9963 - accuracy: 0.3773 - val_loss: 0.9628 - val_accuracy: 0.7692 - 8s/epoch - 2s/step
Epoch 3/200
4/4 - 7s - loss: 0.9901 - accuracy: 0.4727 - val_loss: 0.9580 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 4/200
4/4 - 7s - loss: 0.9841 - accuracy: 0.6091 - val_loss: 0.9525 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 5/200
4/4 - 8s - loss: 0.9782 - accuracy: 0.7182 - val_loss: 0.9462 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 6/200
4/4 - 10s - loss: 0.9751 - accuracy: 0.7318 - val_loss: 0.9389 - val_accuracy: 0.7949 - 10s/epoch - 3s/step
Epoch 7/200
4/4 - 7s - loss: 0.9679 - accuracy: 0.7773 - val_loss: 0.9307 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 8/200
4/4 - 7s - loss: 0.9650 - accuracy: 0.7682 - val_loss: 0.9209 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 9/200
4/4 - 7s - loss: 0.9562 - accuracy: 0.7682 - val_loss: 0.9089 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 10/200
4/4 - 8s - loss: 0.9462 - accuracy: 0.7727 - val_loss: 0.8935 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 11/200
4/4 - 9s - loss: 0.9358 - accuracy: 0.7727 - val_loss: 0.8736 - val_accuracy: 0.7949 - 9s/epoch - 2s/step
Epoch 12/200
4/4 - 7s - loss: 0.9126 - accuracy: 0.7727 - val_loss: 0.8474 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 13/200
4/4 - 7s - loss: 0.8931 - accuracy: 0.7727 - val_loss: 0.8142 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 14/200
4/4 - 8s - loss: 0.8603 - accuracy: 0.7727 - val_loss: 0.7714 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 15/200
4/4 - 8s - loss: 0.8173 - accuracy: 0.7727 - val_loss: 0.7286 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 16/200
4/4 - 8s - loss: 0.7783 - accuracy: 0.7727 - val_loss: 0.6961 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 17/200
4/4 - 7s - loss: 0.7465 - accuracy: 0.7727 - val_loss: 0.6796 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 18/200
4/4 - 8s - loss: 0.7266 - accuracy: 0.7727 - val_loss: 0.6739 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 19/200
4/4 - 8s - loss: 0.7206 - accuracy: 0.7727 - val_loss: 0.6721 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
```

Figure 42: Training the model output 1

```
Epoch 182/200
4/4 - 7s - loss: 0.6812 - accuracy: 0.7727 - val_loss: 0.6367 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 183/200
4/4 - 7s - loss: 0.6809 - accuracy: 0.7727 - val_loss: 0.6364 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 184/200
4/4 - 8s - loss: 0.6807 - accuracy: 0.7727 - val_loss: 0.6362 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 185/200
4/4 - 8s - loss: 0.6804 - accuracy: 0.7727 - val_loss: 0.6359 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 186/200
4/4 - 7s - loss: 0.6801 - accuracy: 0.7727 - val_loss: 0.6356 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 187/200
4/4 - 8s - loss: 0.6799 - accuracy: 0.7727 - val_loss: 0.6354 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 188/200
4/4 - 8s - loss: 0.6797 - accuracy: 0.7727 - val_loss: 0.6351 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 189/200
4/4 - 7s - loss: 0.6794 - accuracy: 0.7727 - val_loss: 0.6349 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 190/200
4/4 - 8s - loss: 0.6791 - accuracy: 0.7727 - val_loss: 0.6346 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 191/200
4/4 - 8s - loss: 0.6788 - accuracy: 0.7727 - val_loss: 0.6344 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 192/200
4/4 - 8s - loss: 0.6786 - accuracy: 0.7727 - val_loss: 0.6341 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 193/200
4/4 - 8s - loss: 0.6783 - accuracy: 0.7727 - val_loss: 0.6338 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 194/200
4/4 - 8s - loss: 0.6781 - accuracy: 0.7727 - val_loss: 0.6336 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 195/200
4/4 - 7s - loss: 0.6778 - accuracy: 0.7727 - val_loss: 0.6333 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 196/200
4/4 - 8s - loss: 0.6775 - accuracy: 0.7727 - val_loss: 0.6331 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 197/200
4/4 - 8s - loss: 0.6773 - accuracy: 0.7727 - val_loss: 0.6328 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 198/200
4/4 - 8s - loss: 0.6770 - accuracy: 0.7727 - val_loss: 0.6325 - val_accuracy: 0.7949 - 8s/epoch - 2s/step
Epoch 199/200
4/4 - 7s - loss: 0.6768 - accuracy: 0.7727 - val_loss: 0.6323 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
Epoch 200/200
4/4 - 7s - loss: 0.6765 - accuracy: 0.7727 - val_loss: 0.6320 - val_accuracy: 0.7949 - 7s/epoch - 2s/step
```

Figure 43: Training the model output2

-this code to download the model with extension .tflite(tensower flow lite) to take it and put in flutter app to integrate between them and make the app to predict and show the result in the interface:

```
#tf-lite
#1
model.save('my_model.h5')

#2
!pip install tensorflow==2.7.0
```

```
#3
import tensorflow as tf
import os

# SavedModel file in the current directory
print(os.getcwd())  # Print the current working directory

saved_model_path = './my_model.h5'  # Specify the correct file path

# Load the Keras model from the SavedModel file
model = tf.keras.models.load_model(saved_model_path)
```

```python
# Convert the Keras model to a TensorFlow Lite model
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

```python
#4
# Save the TensorFlow Lite model to a file and download it
with open('my_model.tflite', 'wb') as f:
  f.write(tflite_model)
```

```python
#5
from google.colab import files

files.download('my_model.tflite')
```

```python
with open('my_file.txt', 'w') as f:
    f.write('1 MS lesions\n')
    f.write('0 normal images\n')

files.download('my_file.txt')
```

Flutter Code:

Main File

```dart
//Import authController.dart file contents and this to manage the
authentication logic using Firebase
import 'package:firebase_auth_app/authController.dart';
import 'package:firebase_auth_app/loginPage.dart';
// Importing the firebase_core.dart file contains the Firebase core SDK
// to initialize firebase in the app
import 'package:firebase_core/firebase_core.dart';
// Importing the material.dart file
// contains the widgets for Material Design
import 'package:flutter/material.dart';
// Importing the get.dart file that contains the Get library
// This library is state managment library for Flutter
import 'package:get/get.dart';
```

```dart
import 'checkAndUpload.dart';
import 'finalResult.dart';

//import 'package:tflite/tflite.dart';

//import 'package:image_labeling/image_labeling.dart';

// main Function is the entry point of app

Future<void> main() async {
  //
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp().then((value) =>
Get.put(AuthController()));
  runApp(const MyApp());
}

// MyAPP is the root widget of app
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    //  GetMaterialApp :the main widget that sets up the application's
theme
    return GetMaterialApp(
      //
      debugShowCheckedModeBanner: false,
      title: 'Daignose MS patients',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: LoginPage(),
    );
  }
}
```

Login page

```dart
//Import authController.dart file contents and this to manage the
authentication logic using Firebase
import 'package:firebase_auth_app/authController.dart';
import 'package:firebase_auth_app/signupPage.dart';
```

```dart
import 'package:firebase_auth_app/welcomePage.dart';
//provides iOS-style widgets for Flutter apps.
import 'package:flutter/cupertino.dart';
//classes for handling gestures on touch-based devices,
import 'package:flutter/gestures.dart';
// Importing the material.dart file
// contains the widgets for Material Design
import 'package:flutter/material.dart';
// to build interface of flutter app
import 'package:flutter/widgets.dart';
// Importing the get.dart file that contains the Get library
// This library is state managment library for Flutter
import 'package:get/get.dart';

class LoginPage extends StatefulWidget {
  const LoginPage({Key? key}) : super(key: key);

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  var emailController = TextEditingController();
  var passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    double w = MediaQuery.of(context).size.width;
    double h = MediaQuery.of(context).size.height;

    return Scaffold(
      backgroundColor: Colors.white,
      body: SingleChildScrollView(
        child: Column(
          children: [
            // container to put logo (image) in the future
            Container(
              width: w,
              height: h * 0.3,
              decoration: BoxDecoration(
                image: DecorationImage(
                  image: AssetImage(" "),
                  fit: BoxFit.cover,
                ),
              ),
```

```
),
SizedBox(
  height: 30,
),

// welcome text
Container(
  width: w,
  margin: EdgeInsets.only(
    left: 20,
    right: 20,
  ),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      const Text(
        "Welcome ",
        style: TextStyle(
          color: Colors.black87,
          fontWeight: FontWeight.bold,
          fontSize: 43.0,
        ),
      ),
      Text(
        "Sign into your account",
        style: TextStyle(
          fontSize: 20.0,
          color: Colors.grey[500],
        ),
      ),
      SizedBox(
        height: 50,
      ),

      // E-mail
      Container(
        decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(40),
            boxShadow: [
              BoxShadow(
                blurRadius: 10,
                offset: const Offset(1, 1),
                color: Colors.grey.withOpacity(0.5),
              ),
```

```
        ]),
    child: TextField(
      controller: emailController,
      decoration: InputDecoration(
        hintText: "Email",
        hintStyle: const TextStyle(
          color: Colors.grey,
        ),
        prefixIcon: const Icon(
          Icons.email,
          color: Color.fromARGB(255, 4, 70, 136),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(40),
          borderSide: const BorderSide(
            color: Colors.transparent,
          ),
        ),
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(40),
          borderSide: const BorderSide(
            color: Colors.transparent,
          ),
        ),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(40),
        ),
      ),
    ),
  ),
  SizedBox(
    height: 20,
  ),

  // password box
  Container(
    decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(40),
        boxShadow: [
          BoxShadow(
            blurRadius: 10,
            offset: Offset(1, 1),
            color: Colors.grey.withOpacity(0.5),
          ),
```

```
                ]),
              child: TextField(
                controller: passwordController,
                obscureText: true,
                decoration: InputDecoration(
                  hintText: "Password",
                  hintStyle: TextStyle(
                    color: Colors.grey,
                  ),
                  prefixIcon: Icon(
                    Icons.password_sharp,
                    color: Color.fromARGB(255, 4, 70, 136),
                  ),
                  focusedBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(40),
                    borderSide: const BorderSide(
                      color: Colors.transparent,
                    ),
                  ),
                  enabledBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(40),
                    borderSide: const BorderSide(
                      color: Colors.transparent,
                    ),
                  ),
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(40),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
      SizedBox(
        height: 10,
      ),

      // Forget password
      Container(
        alignment: Alignment.topRight,
        margin: EdgeInsets.symmetric(horizontal: 30),
        child: const Text(
          "Forgot password?",
          style: TextStyle(
```

```
        color: Colors.grey,
        fontSize: 15,
      ),
    ),
  ),
  SizedBox(
    height: 30,
  ),
  GestureDetector(
    onTap: () {
      AuthController.instance.login(emailController.text.trim(),
          passwordController.text.trim());
    },

    // sign in button
    child: Container(
      width: w * 0.6,
      height: h * 0.08,
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(40),
        image: const DecorationImage(
          image: AssetImage("img/button_col.jpeg"),
          fit: BoxFit.cover,
        ),
      ),
      alignment: Alignment.center,
      child: const Text(
        'Sign in',
        style: TextStyle(
          fontWeight: FontWeight.w600,
          fontSize: 30,
          color: Colors.white,
        ),
      ),
    ),
  ),
  SizedBox(
    height: h * 0.1,
  ),

  // To create account
  RichText(
    text: TextSpan(children: [
      const TextSpan(
        text: 'Don\'t have account? ',
```

```
                  style: TextStyle(
                    color: Colors.grey,
                    fontSize: 18,
                  ),
                ),
                TextSpan(
                  text: 'Create',
                  style: TextStyle(
                    fontSize: 18,
                    color: Colors.grey[700],
                    fontWeight: FontWeight.bold,
                  ),
                  recognizer: TapGestureRecognizer()
                    ..onTap = () {
                      Get.to(() => SignUpPage());
                    },
                ),
              ]),
            )
          ],
        ),
      ),
    );
  }
}
```

Output of this page:

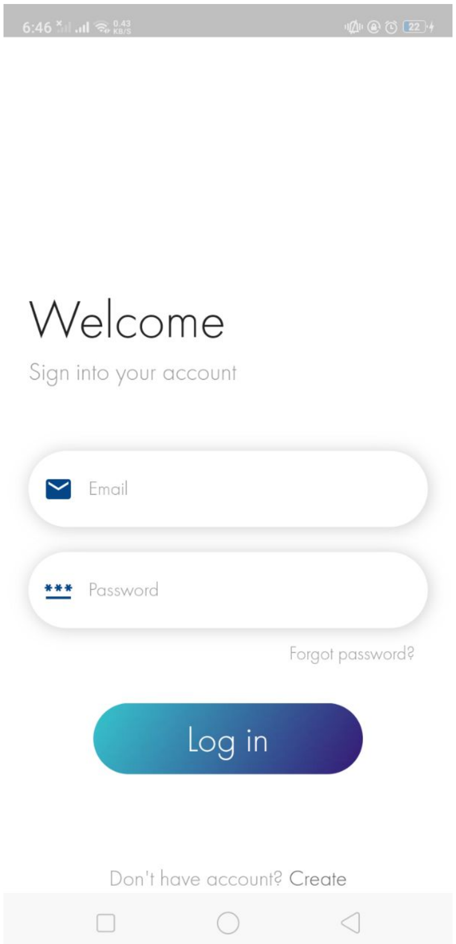


Figure 44: Screenshot of Login page from App

Sign In page

```
import 'package:firebase_auth_app/authController.dart';
import 'package:firebase_auth_app/loginPage.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class SignUpPage extends StatefulWidget {
  const SignUpPage({Key? key}) : super(key: key);

  @override
  State<SignUpPage> createState() => _SignUpPageState();
}

class _SignUpPageState extends State<SignUpPage> {
```

```dart
var emailController = TextEditingController();
var passwordController = TextEditingController();

@override
Widget build(BuildContext context) {
  double w = MediaQuery.of(context).size.width;
  double h = MediaQuery.of(context).size.height;

  return Scaffold(
    backgroundColor: Colors.white,
    body: SingleChildScrollView(
      child: Column(
        children: [
          Container(
            width: w,
            height: h * 0.3,
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage('img/background_col.jpeg'),
                fit: BoxFit.cover,
              ),
            ),
            child: Column(
              children: [
                SizedBox(
                  height: h * 0.148,
                ),
                CircleAvatar(
                  radius: w * 0.15,
                  backgroundColor: Colors.white,
                  backgroundImage: const AssetImage('img/profile.png'),
                )
              ],
            ),
          ),
          SizedBox(
            height: h * 0.08,
          ),
          Container(
            margin: const EdgeInsets.symmetric(
              horizontal: 20,
            ),
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(40),
```

```
          boxShadow: [
            BoxShadow(
              offset: const Offset(1, 1),
              blurRadius: 10,
              color: Colors.grey.withOpacity(0.5),
            ),
          ],
        ),
        child: TextField(
          controller: emailController,
          decoration: InputDecoration(
            hintText: "Email",
            hintStyle: const TextStyle(
              color: Colors.grey,
            ),
            prefixIcon: const Icon(
              Icons.email,
              color: Color.fromARGB(255, 4, 70, 136),
            ),
            enabledBorder: OutlineInputBorder(
              borderRadius: BorderRadius.circular(40),
              borderSide: const BorderSide(
                color: Colors.transparent,
              ),
            ),
            focusedBorder: OutlineInputBorder(
                borderRadius: BorderRadius.circular(40),
                borderSide: const BorderSide(
                  color: Colors.transparent,
                )),
            border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(40)),
          ),
        ),
      ),
      SizedBox(
        height: h * 0.02,
      ),
      Container(
        margin: const EdgeInsets.symmetric(
          horizontal: 20,
        ),
        decoration: BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.circular(40),
```

```
      boxShadow: [
        BoxShadow(
            offset: const Offset(1, 1),
            blurRadius: 10,
            color: Colors.grey.withOpacity(0.5)),
      ],
    ),
    child: TextField(
      controller: passwordController,
      obscureText: true,
      decoration: InputDecoration(
        hintText: "Password",
        hintStyle: const TextStyle(
          color: Colors.grey,
        ),
        prefixIcon: const Icon(
          Icons.password_sharp,
          color: Color.fromARGB(255, 4, 70, 136),
        ),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(40),
        ),
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(40),
          borderSide: const BorderSide(
            color: Colors.transparent,
          ),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(40),
          borderSide: const BorderSide(
            color: Colors.transparent,
          ),
        ),
      ),
    ),
  ),
  SizedBox(
    height: h * 0.02,
  ),
  Container(
    alignment: Alignment.centerRight,
    margin: const EdgeInsets.symmetric(
      horizontal: 25,
    ),
```

```
                child: RichText(
                  text: TextSpan(children: [
                    const TextSpan(
                      text: 'Already have account? ',
                      style: TextStyle(
                        color: Colors.grey,
                        fontSize: 15,
                      ),
                    ),
                    TextSpan(
                      text: 'Log in',
                      style: TextStyle(
                        fontSize: 15,
                        color: Colors.grey[700],
                        fontWeight: FontWeight.bold,
                      ),
                      recognizer: TapGestureRecognizer()
                        ..onTap = () {
                          Get.to(() => const LoginPage());
                        },
                    ),
                  ]),
                ),
              ),
              SizedBox(
                height: h * 0.08,
              ),
              GestureDetector(
                onTap: () {

AuthController.instance.register(emailController.text.trim(),
                      passwordController.text.trim());
                },
                child: Container(
                  height: h * 0.08,
                  width: w * 0.6,
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(40),
                    image: const DecorationImage(
                      image: AssetImage('img/button_col.jpeg'),
                      fit: BoxFit.cover,
                    ),
                  ),
                  alignment: Alignment.center,
                  child: const Text(
```

```
                  'Sign in',
                  style: TextStyle(
                    fontWeight: FontWeight.w600,
                    fontSize: 30,
                    color: Colors.white,
                  ),
                ),
              ),
            ),
            SizedBox(
              height: h * 0.08,
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:

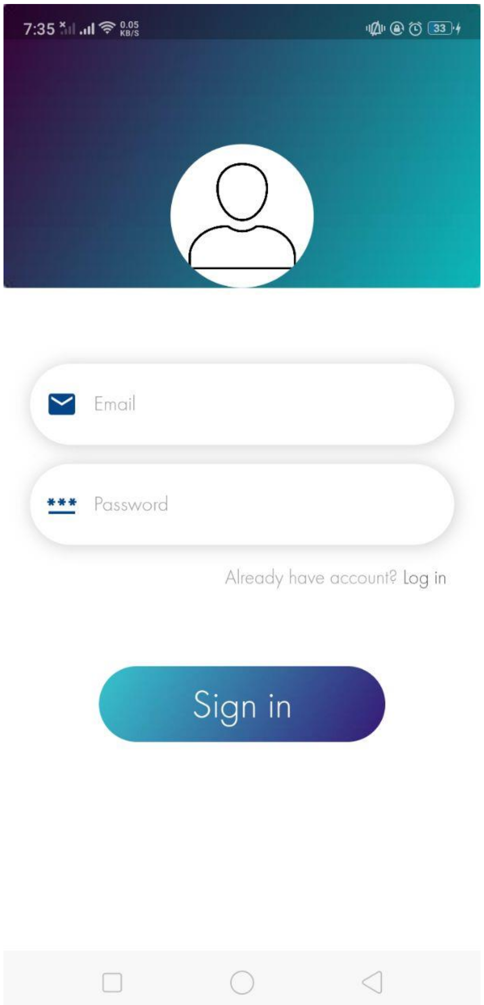


Figure 45: Screenshot of Sign in Page from app

Welcome Page

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_auth_app/authController.dart';
import 'package:firebase_auth_app/scanPage.dart';
import 'package:flutter/material.dart';
import 'dart:ui';

import 'package:get/get.dart';

import 'loginPage.dart';

class WelcomePage extends StatelessWidget {
  String email;
  WelcomePage({Key? key, required this.email}) : super(key: key);

  @override
```

```
Widget build(BuildContext context) {
  double w = MediaQuery.of(context).size.width;
  double h = MediaQuery.of(context).size.height;
  return Scaffold(
    backgroundColor: Colors.white,
    body: Column(
      children: [
        Container(
          width: w,
          height: h * 0.4,
          decoration: const BoxDecoration(
            image: DecorationImage(
              image: AssetImage('img/background_col.jpeg'),
              fit: BoxFit.cover,
            ),
          ),
          child: Column(
            children: [
              SizedBox(
                height: h * 0.21,
              ),
              CircleAvatar(
                radius: w * 0.18,
                backgroundColor: Colors.white,
                backgroundImage: AssetImage('img/profile.png'),
              )
            ],
          ),
        ),
        SizedBox(
          height: h * 0.06,
        ),
        Container(
          width: w,
          margin: const EdgeInsets.only(
            left: 20,
            right: 20,
          ),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              const Text(
                "Welcome",
                style: TextStyle(
                  color: Colors.black87,
```

```
                fontWeight: FontWeight.bold,
                fontSize: 43.0,
              ),
            ),
            Text(
              email,
              style: TextStyle(
                fontSize: 20.0,
                color: Colors.grey[500],
              ),
            ),
          ],
        ),
      ),
      SizedBox(
        height: h * 0.2,
      ),
      GestureDetector(
        onTap: () {
          // to navigate the scanPage (move to scan page when press
the button)
          Get.to(ScanPage());
        },
        child: Container(
          width: w * 0.6,
          height: h * 0.08,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(40),
            image: const DecorationImage(
              image: AssetImage("img/button_col.jpeg"),
              fit: BoxFit.cover,
            ),
          ),
          alignment: Alignment.center,
          child: const Text(
            'Start',
            style: TextStyle(
              fontWeight: FontWeight.w600,
              fontSize: 30,
              color: Colors.white,
            ),
          ),
        ),
      ),
      SizedBox(
```

```
            height: h * 0.02,
          ),
          TextButton(
            onPressed: () {
              // to navigate to the login page when press the button
              Get.offAll(LoginPage());
            },
            child: const Text(
              'Sign out',
              style: TextStyle(
                fontWeight: FontWeight.w600,
                fontSize: 20,
                color: Colors.black87,
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

Output:

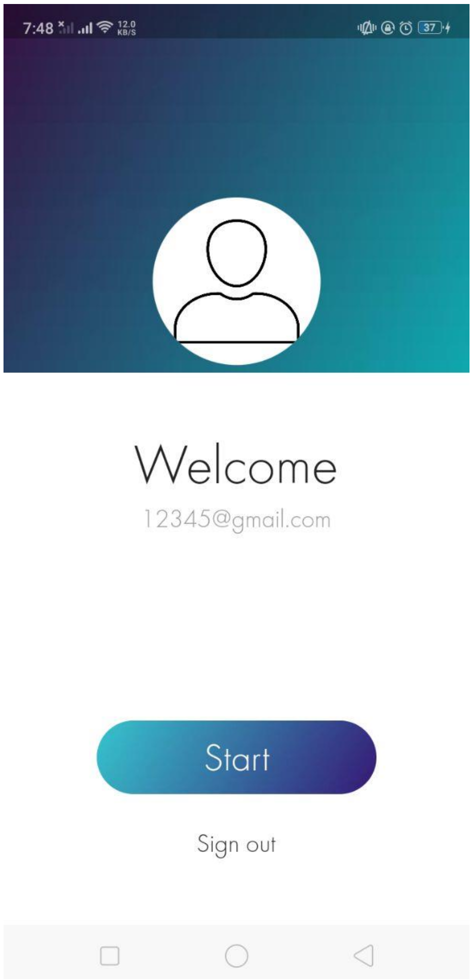


Figure 46: Screenshot of welcome page from app

Scanning page

```
import 'dart:io';
import 'package:flutter/material.dart';

import 'package:firebase_auth_app/authController.dart';
import 'package:firebase_auth_app/signupPage.dart';
import 'package:firebase_auth_app/welcomePage.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';
import 'package:get/get.dart';
import 'checkAndUpload.dart';

class ScanPage extends StatelessWidget {
```

```
  const ScanPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Scanning Page'),
        backgroundColor: Color.fromARGB(248, 7, 62, 125),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Container(
              width: 200,
              height: 200,
              decoration: BoxDecoration(
                color: Colors.grey[200],
                borderRadius: BorderRadius.circular(16),
              ),
              child: GestureDetector(
                onTap: () {
                  // Navigate to the check_and_upload page when the button
is pressed
                  // Navigator.pushNamed(context, '/check_and_upload');
                  Get.to(() => check_and_upload());
                },
                child: Container(
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(40),
                    image: DecorationImage(
                      image: AssetImage("img/button_col.jpeg"),
                      fit: BoxFit.cover,
                    ),
                  ),
                  alignment: Alignment.center,
                  child: const Text(
                    'Scanning',
                    style: TextStyle(
                      fontSize: 30,
                      color: Colors.white,
                    ),
                  ),
                ),
              ),
```

```
                ),
              ],
            ),
          ),
        );
      }
    }
```

Output:

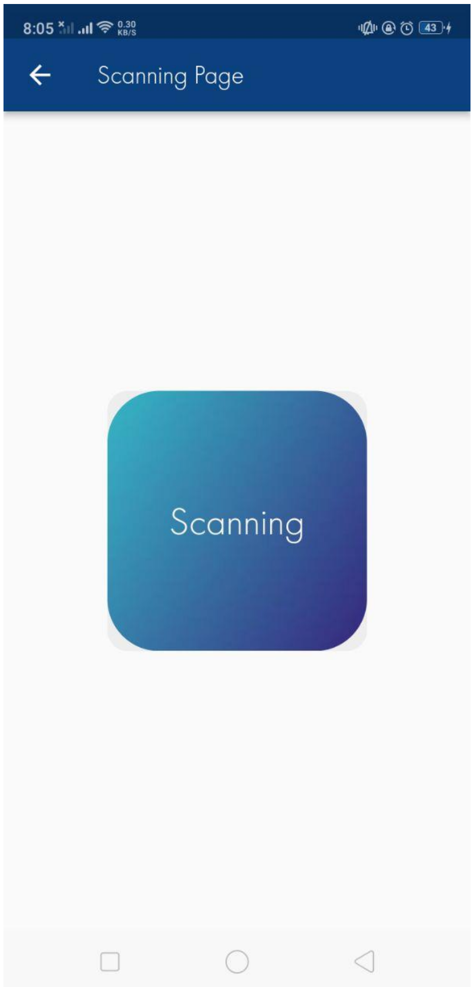


Figure 47: Screenshot of scanning page from App

Check and upload page

```
//Import authController.dart file contents and this to manage the
authentication logic using Firebase
import 'package:firebase_auth_app/authController.dart';
import 'package:firebase_auth_app/loginPage.dart';
```

```dart
// Importing the firebase_core.dart file contains the Firebase core SDK
// to initialize firebase in the app
import 'package:firebase_core/firebase_core.dart';
// Importing the material.dart file
// contains the widgets for Material Design
import 'package:flutter/material.dart';
// Importing the get.dart file that contains the Get library
// This library is state managment library for Flutter
import 'package:get/get.dart';

// to upload image from gallery
import 'package:image_picker/image_picker.dart';
import 'dart:io';

import 'finalResult.dart';

class check_and_upload extends StatefulWidget {
  const check_and_upload({Key? key}) : super(key: key);

  @override
  _check_and_uploadState createState() => _check_and_uploadState();
}

class _check_and_uploadState extends State<check_and_upload> {
  File? _image;
  final picker = ImagePicker();
  String? _result;

  // Load the tflite model when the widget is created
  @override
  // void initState() {
  // super.initState();
  // loadModel();
  //}

  // Function to load the tflite model
  /*
  void loadModel() async {
    await Tflite.loadModel(
      model: 'assets/model.tflite',
      labels: 'assets/labels.txt',
    );
  }

  // Function to apply the tflite model on the selected image
```

```
  Future applyModelOnImage(File file) async {
    var recognitions = await Tflite.runModelOnImage(
      path: file.path,
      numResults: 2,
    );
    setState(() {
      _result = recognitions?.isNotEmpty == true
          ? recognitions![0]['label']
          : 'No Result';
    });
  }
*/
  // Function to select an image from the gallery
  Future getImageFromGallery() async {
    final pickedImage = await picker.pickImage(source:
ImageSource.gallery);
    if (pickedImage != null) {
      final File image = File(pickedImage.path);
      setState(() {
        _image = image;
      });
    }
  }

  // Function to apply the tflite model on the selected image and navigate
to the final result page
  Future navigateToFinalResultPage() async {
    if (_image != null) {
      //  await applyModelOnImage(_image!);
      Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => FinalResult(result: _result ?? 'No
Result')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Upload and check'),
        backgroundColor: Color.fromARGB(248, 7, 62, 125),
      ),
```

```
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Container(
                width: 200,
                height: 200,
                decoration: BoxDecoration(
                  color: Colors.grey[200],
                  borderRadius: BorderRadius.circular(16),
                ),
                child: _image == null
                    ? Center(
                        child: Text(
                          'No image selected.',
                          style: TextStyle(fontSize: 24),
                        ),
                      )
                    : Image.file(
                        _image!,
                        fit: BoxFit.cover,
                      ),
              ),
              SizedBox(height: 16),
              GestureDetector(
                onTap: getImageFromGallery,
                child: Container(
                  width: 200,
                  height: 50,
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(40),
                    image: DecorationImage(
                      image: AssetImage("img/button_col.jpeg"),
                      fit: BoxFit.cover,
                    ),
                  ),
                  alignment: Alignment.center,
                  child: Text(
                    'Upload Image',
                    style: TextStyle(fontSize: 22, color: Colors.white),
                  ),
                ),
              ),
              SizedBox(height: 16),
              GestureDetector(
```

```
              onTap: navigateToFinalResultPage,
              child: Container(
                width: 200,
                height: 50,
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(40),
                  image: DecorationImage(
                    image: AssetImage("img/button_col.jpeg"),
                    fit: BoxFit.cover,
                  ),
                ),
                alignment: Alignment.center,
                child: Text(
                  'Ready to Check',
                  style: TextStyle(fontSize: 22, color: Colors.white),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:

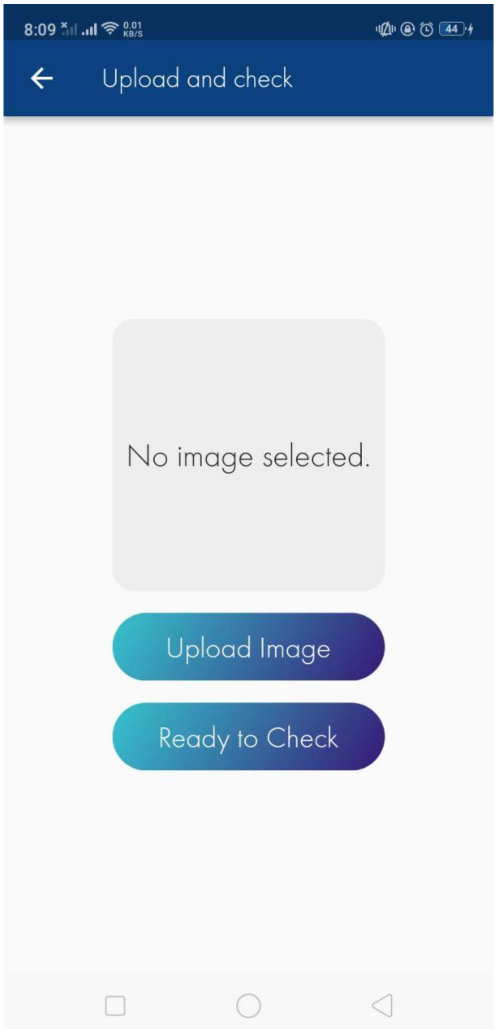


Figure 48: Screenshot of Check and upload image

Final Result page

```
import 'dart:io';
import 'dart:typed_data';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:tflite_flutter/tflite_flutter.dart' as tfl;
import 'package:image/image.dart' as img;

late tfl.Interpreter interpreter;
late String result;
List<double> outputList = [];

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
```

```
  try {
    interpreter = await tfl.Interpreter.fromAsset('model_unquant.tflite');
    print('Interpreting model done successfully ✓');
  } catch (e) {
    print('Error while interpreting model !!');
  }
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key}) : super(key: key);

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  File? _image;

  late Object output;

  Future<void> _pickImage(ImageSource source) async {
    final image = await ImagePicker().getImage(source: source);
    if (image == null) return;
    // to show the result
    result = await predictImage(File(image.path));
    outputList = convertOutputToList(result);
    //assignOutputToVariables(outputList);
    useOutputList(outputList);

    print(result);

    setState(() {
      _image = File(image.path);
```

```
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Final Result'),
        backgroundColor: const Color.fromARGB(248, 7, 62, 125),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            if (_image != null) ...[
              Container(
                height: 200, // set the height and width to the same value
                width: 200,
                child: Image.file(_image!),
              ),
              const SizedBox(height: 20),
              Text(
                //outputList.join(", "),
                useOutputList(outputList),
                style: const TextStyle(fontSize: 18),
              ),
              const SizedBox(height: 20),
            ],
            ElevatedButton(
              child: const Text('Pick Image'),
              style: ElevatedButton.styleFrom(
                backgroundColor: const Color.fromARGB(248, 7, 62, 125),
              ),
              onPressed: () => _pickImage(ImageSource.gallery),
            ),
          ],
        ),
      ),
    );
  }

  Future<String> predictImage(File imageFile) async {
    try {
      // Load the image and resize it to the expected input size
```

```
      // decodeImage() function to decode image (convert image from bytes
and return it as object)
      img.Image originalImage = img.decodeImage(await
imageFile.readAsBytes())!;
      // get the size of input in the loaded TFlite model
      int inputSize = interpreter.getInputTensor(0).shape[1];
      // change the size of input image to use it in Tflite model
      img.Image resizedImage =
          img.copyResize(originalImage, width: inputSize, height:
inputSize);

      // Convert the image to grayscale
      // to make the input image in tflite in grayScale
      img.Image grayscaleImage = img.grayscale(resizedImage);

      // Duplicate the grayscale channel to create a 3-channel image
      img.Image rgbImage = img.Image.from(grayscaleImage);

      // Normalize the image pixels (assuming pixel values in range [0,
255])
      var imageBytes = rgbImage.getBytes();
      var normalizedPixels =
          imageBytes.map((pixelValue) => pixelValue / 255.0).toList();

      // Create a 4-dimensional input tensor (assuming 3-channel image)
      var input = Float32List(inputSize * inputSize * 3)
          .reshape([1, inputSize, inputSize, 3]);
      for (int i = 0; i < inputSize * inputSize; ++i) {
        input[0][i ~/ inputSize][i % inputSize][0] = normalizedPixels[i *
3];
        input[0][i ~/ inputSize][i % inputSize][1] =
            normalizedPixels[i * 3 + 1];
        input[0][i ~/ inputSize][i % inputSize][2] =
            normalizedPixels[i * 3 + 2];
      }

      // Run the interpreter
      final inputShape = interpreter.getInputTensor(0).shape;
      if (inputShape[0] != 1) {
        throw Exception(
            'Invalid input batch size ${inputShape[0]}, expected 1.');
      }
      final outputShape = interpreter.getOutputTensor(0).shape;
      var output =
```

```
          Float32List(outputShape.reduce((a, b) => a *
b)).reshape(outputShape);
      interpreter.run(input, output);

      return output.toString();
    } catch (e) {
      print('Error while predicting image: $e');
      return 'Error';
    }
  }

  List<double> convertOutputToList(String output) {
    List<String> outputStrings = output
        .replaceAll('[', '') // remove brackets
        .replaceAll(']', '')
        .split(', '); // split by comma and space
    return outputStrings.map((s) => double.parse(s)).toList();
  }

  String useOutputList(List<double> outputList) {
    double variable1 = outputList[0];
    double variable2 = outputList[1];
    // double variable3 = outputList[2];
    //double variable4 = outputList[3];

    if (variable1 > 0.25) {
      return 'MS patient';
    } else {
      return 'Not MS patient';
    }
  }
}
```

Output:

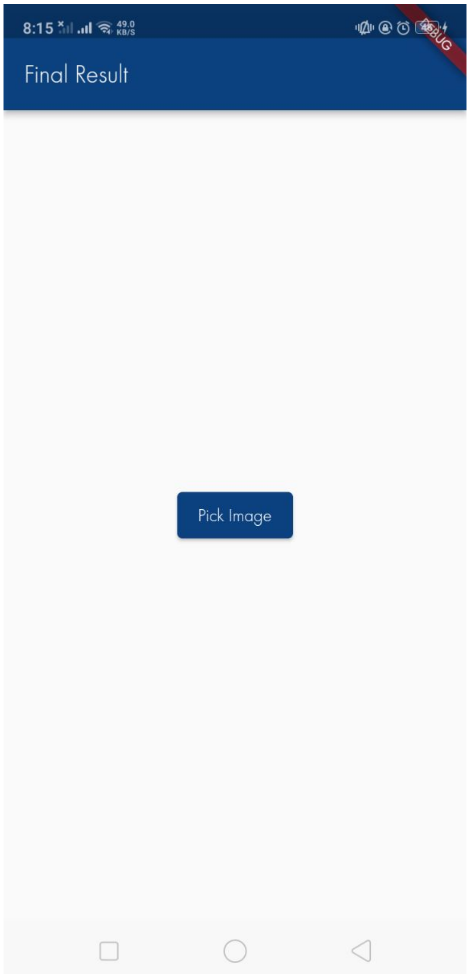


Figure 49: Screenshot of final result page

Testing
Model

Testing the prediction step

```
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array

# load the image
#1
test_image = load_img('/content/drive/MyDrive/project
notes/Dataset_Diagnose_MS/Yes_MS_lesions/10-image.0013 H .jpg',
target_size = (128,128))
test_image
```

Output:

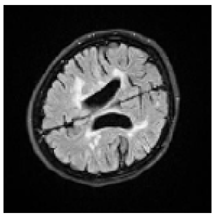


Figure 50: Prediction output 1

```
test_image = img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
resulty = model.predict(test_image)

output = np.round(resulty).astype(int)
output
```

output:

```
1/1 [==============================] - 0s 30ms/step
array([[1]])
```

Figure 51: Prediction output 2

Chapter 5

Results and Discussion

Results of

Model

Summary of model

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 128, 128, 32)      2432

 max_pooling2d_12 (MaxPoolin  (None, 64, 64, 32)       0
 g2D)

 conv2d_13 (Conv2D)          (None, 64, 64, 64)        18496

 max_pooling2d_13 (MaxPoolin  (None, 32, 32, 64)       0
 g2D)

 conv2d_14 (Conv2D)          (None, 32, 32, 128)       73856

 max_pooling2d_14 (MaxPoolin  (None, 16, 16, 128)      0
 g2D)

 conv2d_15 (Conv2D)          (None, 16, 16, 128)       147584

 max_pooling2d_15 (MaxPoolin  (None, 8, 8, 128)        0
 g2D)

 flatten_3 (Flatten)         (None, 8192)              0

 dropout_3 (Dropout)         (None, 8192)              0

 dense_6 (Dense)             (None, 256)               2097408

 dense_7 (Dense)             (None, 1)                 257

=================================================================
Total params: 2,340,033
Trainable params: 2,340,033
Non-trainable params: 0
_________________________________________________________________
```

Figure 52: Summary of model with details

In the summary of model it shows the archticture of CNN model, the model is a sequential model, which means it is a linear stack of layers, it includes 4 convolutional layers, between each other max pooling layer to reduce the spatial dimensions (width and height) of the feature maps produced by convolutional layers, includes the flatten layer which flattens the output of the previous max pooling layer into a 1D array and two dense layer are responsible for learning complex, non-linear relationships between the inputs and the outputs.

The total number of trainable parameters in the model is 2,340,033.

Accuracy

```
Training accuracy: 0.7727272510528564
Validation accuracy: 0.7948718070983887
Test accuracy: 0.7948718070983887
```

Figure 53: Accuracy of CNN model

Flutter App

- Login page

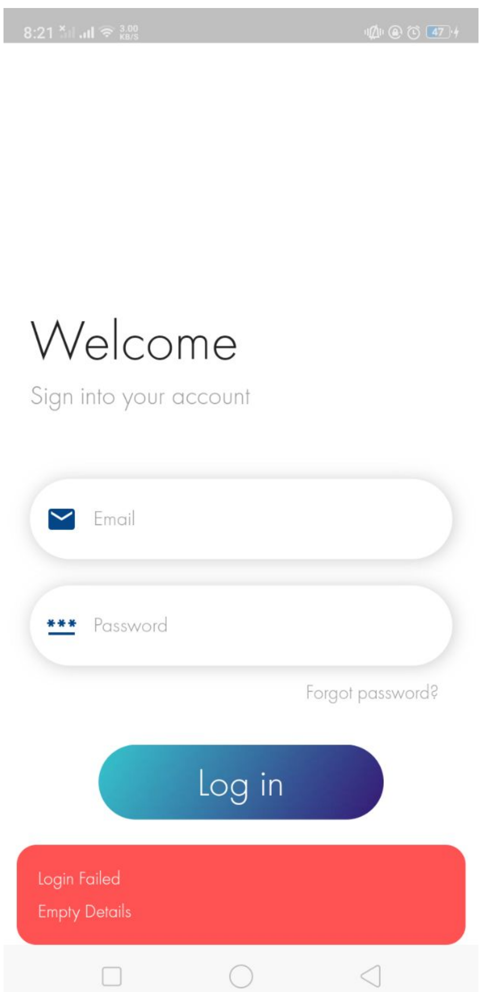


Figure 54: Error message when email and password labels are empty.

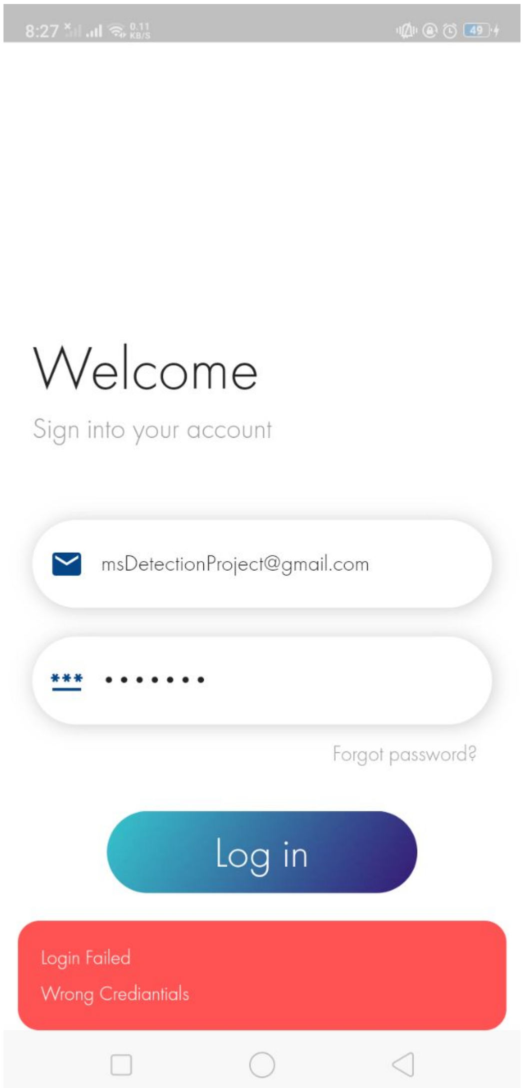


Figure 55: Screenshot of error message when email and password is not found

- Sign In page

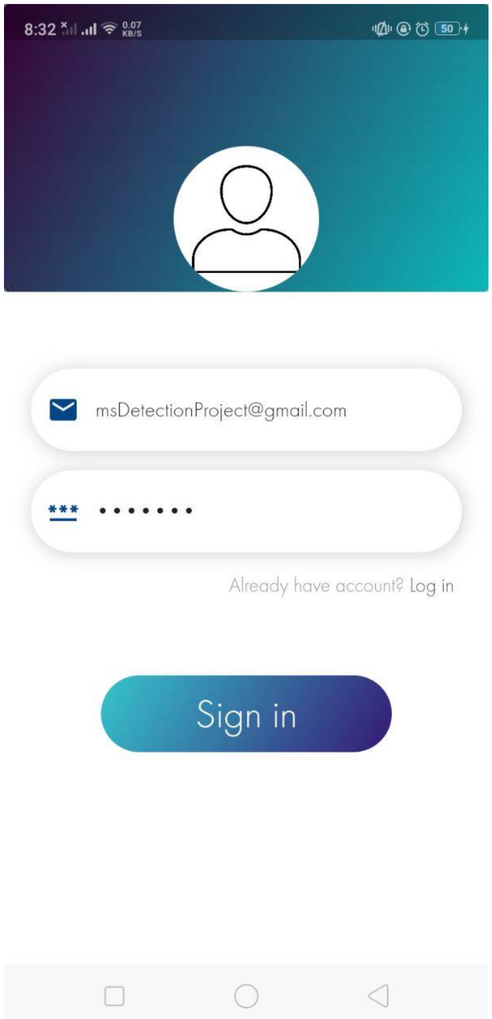


Figure 56: Screenshot of create account in sign in page.

- Check and Upload page

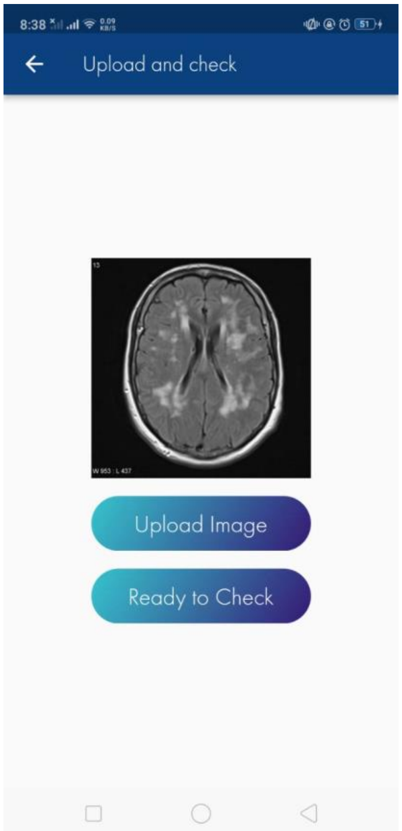


Figure 57: Screenshot of image display in interface of app when click on the upload image button.

- Final Result page

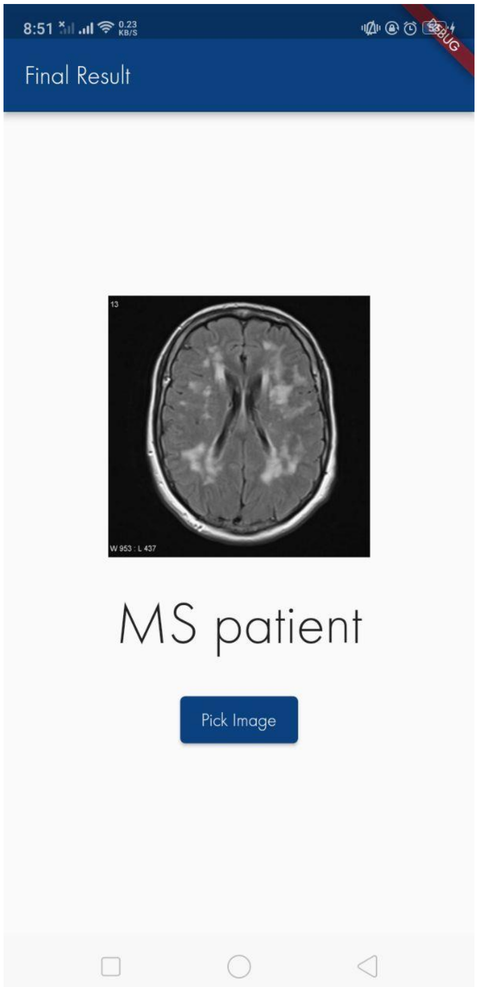


Figure 58: Screenshot of Test case 1: ‘MS patient’ image

Another test case

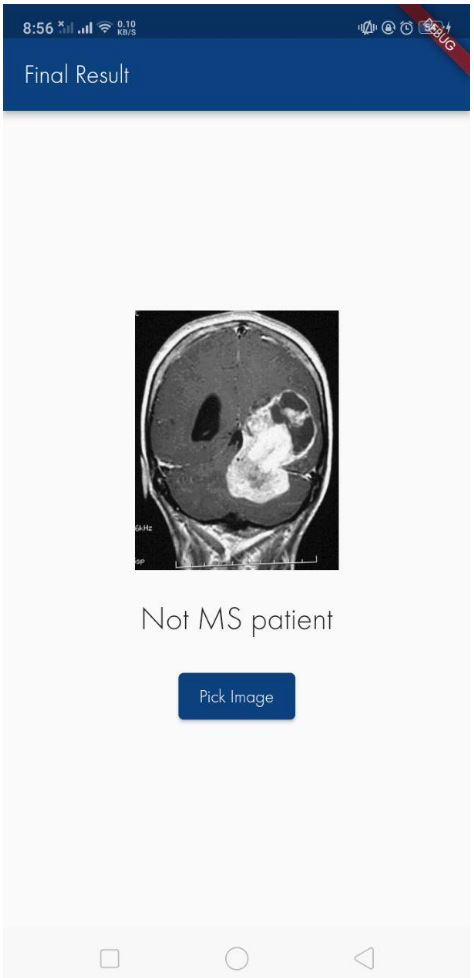


Figure 59: Screenshot of test case 2: ‘Not MS patient’

Chapter 6

Future work and The conclusion

Future work

I am currently working on publishing this project as scientific research to help in the development of systems that aid in detecting and predicting multiple sclerosis and making the diagnosis of this disease more accurate and easier.

Regarding the model:

- The code should be developed to be able to detect and identify the disease from 3D magnetic resonance imaging (MRI) or NIfTI files, not just from 2D images.
- After identifying that the person is sick, the severity of the disease should be determined according to its risk and progression.
- Predicting the patient's condition and identifying the developments that the patient may reach in the future.

As for the development in the application:

- It should support more user features such as:
  . Allowing the user(patient) to save their medical report in the application's database.
  . Allowing the user(patient) to send the report to another user(doctor) and allowing the doctor to input and send their notes on this report.
- Sending alerts to the doctor and notifying them of critical cases.

Conclusion

In conclusion, the development of a mobile app for the classification of MS lesions is an important contribution to the field of computer science and healthcare. The aim of this project was to provide a tool for healthcare professionals and any user want to know if his/her MRI include MS lesions or not, using a mobile app easily and fast. Through extensive research and development, we were able to create a mobile app that achieved this goal.

The app uses advanced image processing techniques and deep learning models to analyze MRI images and classify MS lesions with a high level of accuracy.

References

alokesh985, 2022. *Introduction to Support Vector Machines (SVM), geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm/
[Accessed 2022].

Chalmer, T.A., Baggesen, L.M. et al, 2018. *Early versus later treatment start in multiple sclerosis: a register-based cohort study. European Journal of Neurology.* [Online]
Available at: https://onlinelibrary.wiley.com/doi/abs/10.1111/ene.13692
[Accessed 2018].

Chaudhary, L., n.d. *Agile in SDLC.* [Online]
Available at: https://www.educba.com/agile-in-sdlc/
[Accessed 2023].

Clare Walton, Rachel King, et al, 2020. *Rising prevalence of multiple sclerosis worldwide: Insights from the Atlas of MS, third edition, Sage journals.* [Online]
Available at: https://journals.sagepub.com/doi/epub/10.1177/1352458520970841
[Accessed 14 12 2020].

Franciosi, A., 2020. *The Agile Development Life Cycle.* [Online]
Available at: https://www.objectedge.com/blog/the-agile-development-life-cycle

Gurucharan, M., 2022. *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network.* [Online]
Available at: https://www.upgrad.com/blog/basic-cnn-architecture/
[Accessed 2023].

james le, j. p., 2022. *The Top 10 Machine Learning Algorithms Every Beginner Should Know.* [Online]
Available at: https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies

Khanal, M. P., 2020. *Effective Detection of Brain Tumour on MRI Images Using Optimization Based, International Journal of Scientific & Technology Research.* [Online]
Available at:
https://www.researchgate.net/publication/342833582_Effective_Detection_Of_Brain_Tumour_On_MRI_Images_Using_Optimization_Based_Segmentation_Techniques

Lilla Bonanno, N. M. e. a., 2021. *Clinical Imaging(Multiple Sclerosis lesions detection by a hybrid).* [Online]
Available at: https://www.elsevier.com/locate/clinimag

M. Bilello, M. Arkuszewski, P. Nucifora et al , 2013. *Multiple Sclerosis: Identification of Temporal Changes in Brain Lesions with Computer-Assisted Detection Software, Sage*

journals. [Online]
Available at:
https://journals.sagepub.com/doi/full/10.1177/197140091302600202?casa_token=wQ9Ux2Yeym4AAAAA%3A9EkJkiYTIUhP75YWgb0CBrlroMN1q9KoBq6vcKUUyw1ls18injfnJ3ogsH4MYWUDBf4gcMxTPA6B

Micael S. Couceiro, Rui P. Rocha et al , 2021. *Introducing the fractional-order Darwinian PSO, Springer.* [Online]
Available at: https://link.springer.com/article/10.1007/s11760-012-0316-2

N Durga Indira, e. a., 2020. *Brain Tumor Detection from MRI Images Using Optimization Segmentation Techniques.* [Online]
Available at:
https://www.researchgate.net/publication/344491298_Brain_Tumor_Detection_from_MRI_Images_Using_Optimization_Segmentation_Techniques

P.Roca et al, 2020. *Artificial intelligence to predict clinical disability in patients with multiple sclerosis using FLAIR MRI.* [Online]
Available at:
https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=0CAQQw7AJahcKEwiw2MyZ68v6AhUAAAAAHQAAAAAQAg&url=https%3A%2F%2Fwww.sciencedirect.com%2Fscience%2Farticle%2Fpii%2FS2211568420301558&psig=AOvVaw2LTkxcZL-tnidt2YbUM7cm&ust=16651

Pal, S. K., n.d. *Software Engineering | Classical Waterfall Model.* [Online]
Available at: https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/
[Accessed 4 4 2023].

Sharawy, P., 2020. *MS international federation.* [Online]
Available at: https://www.atlasofms.org/map/egypt/epidemiology/number-of-people-with-ms
[Accessed 2020].

singh, H., 2019. *mage Processing Using Machine Learning.* [Online]
Available at: https://www.oreilly.com/library/view/practical-machine-learning/9781484241493/html/471189_1_En_5_Chapter.xhtml

tabmir, 2022. *SIFT Interest Point Detector Using Python – OpenCV, geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/sift-interest-point-detector-using-python-opencv/
[Accessed 2022].

tabmir, 2023. *SIFT Interest Point Detector Using Python – OpenCV.* [Online]
Available at: https://www.geeksforgeeks.org/sift-interest-point-detector-using-python-

opencv/
[Accessed 2023].

Wadhah Ayadi, Wajdi Elhamzi et al, 2021. *Deep CNN for Brain Tumor Classification, springer.* [Online]
Available at: https://link.springer.com/article/10.1007/s11063-020-10398-2

Wallin, M.T., Culpepper, W.J., et al, 2016. *Global, regional, and national burden of multiple sclerosis 1990–2016: a systematic analysis for the Global Burden of Disease Study 2016. The Lancet Neurology.* [Online]
Available at: https://www.sciencedirect.com/science/article/pii/S1474442218304435
[Accessed 3 2019].

ZHUOQIAN YANG, TINGTING DAN, etal , 2018. *Multi-temporal Remote Sensing Image.* [Online]
Available at: https://www.researchgate.net/publication/326213864_Multi-Temporal_Remote_Sensing_Image_Registration_Using_Deep_Convolutional_Features

Number of words: 11053 words