PARKINSON'S DISEASE PREDICTION

A PROJECT REPORT

Submitted by

GIRISHUN KUMAR R	**113119UG03027**
KISHORE M	**113119UG03050**
SRI NANDHISH KUMAR S	**113119UG03104**
SHRIRAM A	**113119UG03099**

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VEL TECH MULTI TECH Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE, ALAMATHI ROAD, AVADI, CHENNAI-62

ANNA UNIVERSITY, CHENNAI 600 025.

MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report of title "**PARKINSON'S DISEASE DETECTION IN DAILY LIFE**" is the bonafide work of **GIRISHUN KUMAR R (113119UG03027), KISHORE M (113119UG03050), SRI NANDHISH KUMAR S (113119UG03104), SHRIRAM A (113119UG03099)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Dr.R.Saravanan, B.E, M.E(CSE)., Ph.D.	**Mr.V. Nehru,B.TECH(CSE),M.Tech(CSE)**
PROFESSOR,	ASSISTANT PROFESSOR,
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College,	Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College,
Avadi, Chennai-600 062	Avadi, Chennai-600 062

CERTIFICATE FOR EVALUATION

This is to certify that the project entitled “**PARKINSON’S DISEASE DETECTION IN DAILY LIFE**” is the bonafide record of work done by following students to carry out the project work under our guidance during the year 2022-2023 in partial fulfillment for the award of Bachelor of Engineering degree in Computer Science and Engineering conducted by Anna University, Chennai.

GIRISHUN KUMAR R	**(113119UG03027)**
KISHORE M	**(113119UG03050)**
SRI NANDHISH KUMAR S	**(113119UG03104)**
SHRIRAM A	**(113119UG03099)**

This project report was submitted for viva voce held on __________
At Vel Tech Multi Tech Dr. Rangarajan and Dr.Sakunthala Engineering College.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We wish to express our sincere thanks to the Almighty and the people who extended their help during the course of our work. We are greatly and profoundly thankful to our honourable Chairman, Col. Prof.Vel. Shri Dr.R.Rangarajan B.E.(ELEC), B.E. (MECH), M.S.(AUTO)., D.Sc., & Vice Chairman, Dr.Mrs.Sakunthala Rangarajan M.B.B.S., for facilitating us with this opportunity. We also record our sincere thanks to our honorable Principal, Dr.V.Rajamani M.E.,Ph.D., for his kind support to take up this project and complete it successfully. We would like to express our special thanks to our Head of the Department, Dr.R.Saravanan, B.E, M.E(CSE)., Ph.D. Department of Computer Science and Engineering and our project supervisor Dr.V.Nehru . for their moral support by taking keen interest in our project work and guided us all along, till the completion of our project work and also by providing with all the necessary information required for developing a good system with successful completion of the same. Further, the acknowledgement would be incomplete if we would not mention a word of thanks to our most beloved Parents for their continuous support and encouragement all the way through the course that has led us to pursue the degree and confidently complete the project work.

(GIRISHUN KUMAR R) (KISHORE M) (SRI NANDHISH KUMAR S) (SHRIRAM A)

ABSTRACT

Early classification Parkinson’s disease from magnetic resonance imaging (MRI) plays an important role in the diagnosis of such diseases. There are many diagnostic imaging methods used to identify Parkinson’s disease. MRI is commonly used for such tasks because of its unmatched image quality. The relevance of artificial intelligence (AI) in the form of deep learning (DL) has revolutionized new methods of automated medical image diagnosis. This study aimed to develop a robust and efficient method based on transfer learning technique for classifying Parkinson’s disease using MRI. In this article, the popular deep learning architectures are utilized to develop Parkinson’s disease diagnostic systems. The pre-trained models such as Xception, NasNet Large, DenseNet121 and InceptionResNetV2 are used to extract the deep features from brain MRI. The experiment was performed using two benchmark datasets that are openly accessible from the web. Images from the dataset were first cropped, preprocessed, and augmented for accurate and fast training. The performance of the transfer learning models is evaluated using performance metrics such as accuracy, sensitivity, precision, specificity, and F1-score. From the experimental results, our proposed CNN model based on the Xception architecture using ADAM optimizer. The proposed method is superior to the existing literature, indicating that it can be used to classify Parkinson’s diseases quickly and accurately.

KEYWORDS: MRI Scan Report, Tensorflow, Mediapipe, Opencv, Nvidia CNN, Nvidia CUDA, Pytorch, YOLO v5.

TABLE OF CONTENTS

LIST OF FIGURE

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The aging of today's society is associated with an increasing number of patients suffering from neurodegenerative disorders. One of these disorders is Parkinson's disease (PD), and current estimates indicate that the number of people with PD will rise more than twofold, from 4 million in 2005 to 9 million by 2030 . The clinical presentations of PD include progressively slowing movements, limb rigidity, restremor, and posture instability . Unfortunately, even those patients who receive dopaminergic treatment or deep brain stimulation still deteriorate with increasing age, and their mortality rate is two- to threefold higher than that of the general population. Therefore, recognizing PD in its early stage is critical for initiating proper treatments to decrease morbidity and ease the medical burden in the elderly The clinical severity of PD can be divided into five stages, called the Hoehn-Yahr Stages I–V . In Stage I, the patients experience unilateral symptoms, such as asymmetrical gait or hand swing; in Stage II, the disease influences are bilateral and the patient's stability degrades; in Stage III, the disease affects the central reflex mechanism, and the patient tends to fall because of trunk instability; in Stage IV, the patient needs a wheelchair and other assistive devices; and in Stage V, the patient is wheelchair bound or even bedridden. Patients with PD can be classified as having early-stage or advanced-stage disease. In its early stages, denoted in this paper as Early PD and defined as Hoehn-Yahr Stage ≤2, the symptoms include asymmetrical movement reduction of one limb, asymmetrical hand movements, and shuffling when walking, with a preserved posture reflex. In the advanced stages, denoted here as Adv PD and defined as Hoehn-Yahr Stage >2, the symptoms are more progressed and include postural reflex losses, festinating gaits that cause walking instability, and increased risk of falling. However, early detection of PD is challenging because the normal aging population might also exhibit progressive gait slowness, termed senile gait, due to joint osteoarthritis or sarcopenia. Therefore, the aim of the present study was to develop a neural network model that could help physicians recognize the PD gait based on motion characteristics occurring during walking. The model would also facilitate monitoring of the PD disease severity stages for appropriate medication adjustment and intervention. Advances in technology are now improving the early, timely, and accurate diagnosis of PD, especially when machine-learning techniques are applied

1.2. OBJECTIVE

Anxiety and depression are considered as one of the earliest symptoms of PD. Although directly measuring these symptoms is hard, it is possible to infer the conditions of mental health by monitoring smartphone usage. Given an example, we can monitor the time of smartphone usage during a day to predict the risk of depression and monitor the movement of a user during and before the calling to predict the risk of anxiety.

1.3. EXISTING SYSTEM

Early detection of PD enables timely initiation of therapeutic management that decreases morbidity. However, correct recognition of PD, especially in early-stage disease, is challenging because the aging population, which has a high PD prevalence, also commonly exhibits progressive gait slowness due to other disorders, such as joint osteoarthritis or sarcopenia. Therefore, developing a reliable and objective method is crucial for differentiating PD gait characteristics from those of the normal elderly. The aim of this study was to develop neural network models that could use the participants' motion data during walking to identify PD. We recruited 32 drug-naïve PD patients with variable disease severity and 16 age/sex-matched healthy controls, and we measured their motions using inertial measurement unit (IMU) sensors. The IMU data were used to develop neural network models that could identify patients with advanced-stage PD with an average accuracy of 92.72% in validation processes. The models also differentiated patients with early-stage PD from normal elderly subjects with an accuracy of 99.67%. Another independent group of participants recruited to test the developed models confirmed the successful discrimination of PD-affected from healthy elderly, as well as patients at different severity stages. Our results provide support for early diagnosis and disease severity monitoring in patients with PD.

1.4 .PROPOSED SYSTEM

The performance of the classification model is also biased nowadays. claimed that they achieved an accuracy of 99% and 97.5% separately when employing sustained vowels into PD detection, pointing out that the proposed validation approach is speaker-dependent. Although the samples in training and testing set a contrast with each other, they can correspond to the same person. This validation method can lead to an over-optimistic result. In the experiments of claims this accuracy can even decrease to 60% if the validation process. We will implement this program as a virtual camera on a device, allowing us to use the sign language translator to translate hand gestures in real-time feeds from the primary camera and output the translated video with subtitles in the virtual camera of OBS software. By doing so, our proposed system will enable effective communication between hearing-impaired individuals and those who do not understand sign language. To evaluate the performance of our proposed system, we will conduct several experiments to compare its accuracy and response time with other state-of-the-art sign language translation systems. The results of these experiments will help us further improve our system to achieve better performance. In conclusion, our proposed work aims to develop a real time sign language translator using YOLOv5 and PyTorch with the aid of the Nvidia CUDA toolkit, 3 implemented as a virtual camera on a device. The system will enable effective communication between hearing impaired individuals and others who do not understand sign language. The proposed system's performance will be evaluated through experiments, and the results will be used to further improve the system.

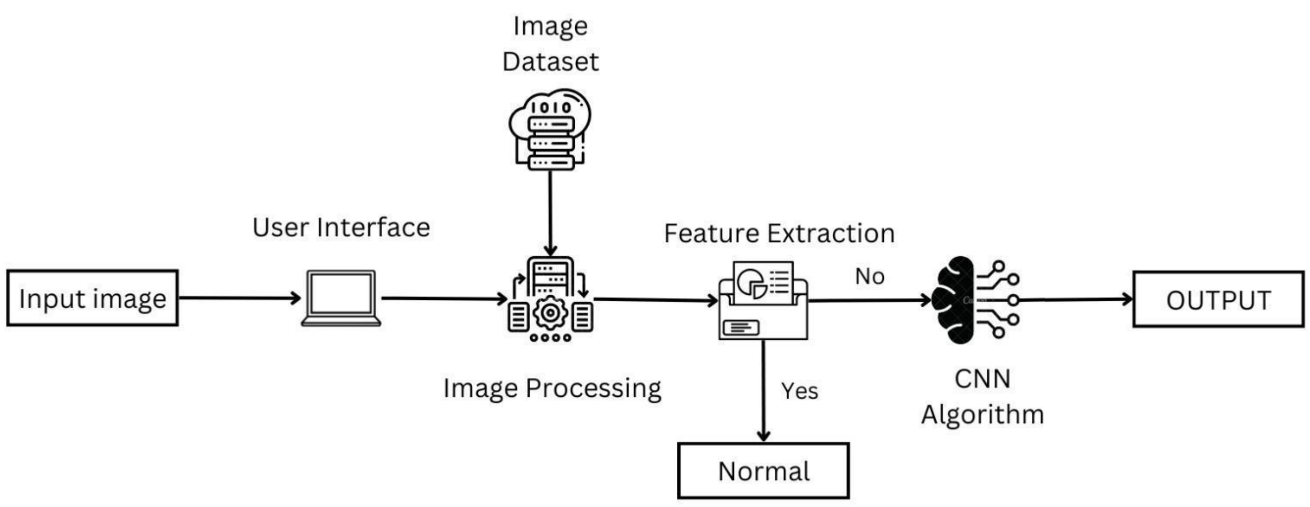


FIG 1.1 Proposed System Architecture

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The aging of today's society is associated with an increasing number of patients suffering from neurodegenerative disorders. One of these disorders is Parkinson's disease (PD), and current estimates indicate that the number of people with PD will rise more than twofold, from 4 million in 2005 to 9 million by 2030 . The clinical presentations of PD include progressively slowing movements, limb rigidity, restremor, and posture instability . Unfortunately, even those patients who receive dopaminergic treatment or deep brain stimulation still deteriorate with increasing age, and their mortality rate is two- to threefold higher than that of the general population. Therefore, recognizing PD in its early stage is critical for initiating proper treatments to decrease morbidity and ease the medical burden in the elderly The clinical severity of PD can be divided into five stages, called the Hoehn-Yahr Stages I–V . In Stage I, the patients experience unilateral symptoms, such as asymmetrical gait or hand swing; in Stage II, the disease influences are bilateral and the patient's stability degrades; in Stage III, the disease affects the central reflex mechanism, and the patient tends to fall because of trunk instability; in Stage IV, the patient needs a wheelchair and other assistive devices; and in Stage V, the patient is wheelchair bound or even bedridden. Patients with PD can be classified as having early-stage or advanced-stage disease. In its early stages, denoted in this paper as Early PD and defined as Hoehn-Yahr Stage ≤2, the symptoms include asymmetrical movement reduction of one limb, asymmetrical hand movements, and shuffling when walking, with a preserved posture reflex. In the advanced stages, denoted here as Adv PD and defined as Hoehn-Yahr Stage >2, the symptoms are more progressed and include postural reflex losses, festinating gaits that cause walking instability, and increased risk of falling. However, early detection of PD is challenging because the normal aging population might also exhibit progressive gait slowness, termed senile gait, due to joint osteoarthritis or sarcopenia. Therefore, the aim of the present study was to develop a neural network model that could help physicians recognize the PD gait based on motion characteristics occurring during walking. The model would also facilitate monitoring of the PD disease severity stages for appropriate medication adjustment and intervention. Advances in technology are now improving the early, timely, and accurate diagnosis of PD, especially when machine-learning techniques are applied.

Aspects of dopaminergic function suggest that dopamine can be involved in different forms of motivation characterized by neuroanatomical specificity and distinct signaling modes, thereby accounting for the varied deficits stemming from dopamine dysfunction. Paired with a stereotypical disease progression in PD from dorsolateral to ventromedial degeneration of midbrain and striatum function, some specificity in the effects of PD on motivation should be expected and these effects are likely to change with disease progression. As outlined above, important progress has been made in under-standing distinctions between different forms of motivation and learning based on computational and neurobiological models. In contrast, psychological and personality level theories targeting motivation have not generally been sensitive to such distinctions. Instead, broad measures of apathy and depression have been used to demonstrate differences in PD patients, but these measures may not always have the resolution to provide tighter links across neural, computational and psychological levels of analysis. Linking self-report measures of motivation with experimental tasks that have been characterized in terms of their computational or neural underpinnings could provide the initial steps needed to bridge across levels of analysis. In order to begin understanding whether dopaminergic dysfunction exerts independent influences on learning and motivation, we sought to determine whether differences in regulatory mode motivation are linked to those specific learning deficits that are associated with dopaminergic dysfunction in the striatum. We reasoned that if learning deficits associated with striatal dopamine dysfunction were associated with motivational mode scores, this would point to a link between striatal dopamine dysfunction and motivational changes.

2.2 Timothy Wroge, Yasin Serdar Özkanca., Parkinson’s Disease Diagnosis Using Machine Learning and Voice,2019

Abstract: Biomarkers derived from human voice can offer insight into neurological disorders, such as Parkinson’s disease (PD), because of their underlying cognitive and neuromuscular function. PD is a progressive neurodegenerative disorder that affects about one million people in the United

States, with approximately sixty thousand new clinical diagnoses made each year. Historically, PD has been difficult to quantify and doctors have tended to focus on some symptoms while ignoring others, relying primarily on subjective rating scales. Due to the decrease in motor control that is the hallmark of the disease, voice can be used as a means to detect and diagnose PD. With advancements in technology and the prevalence of audio collecting devices in daily lives, reliable models that can translate this audio data into a diagnostic tool for healthcare professionals would potentially provide diagnoses that are cheaper and more accurate. We provide evidence to validate this concept here using a voice dataset collected from people with and without PD. This paper explores the effectiveness of using supervised classification algorithms, such as deep neural networks, to accurately diagnose individuals with the disease. Our peak accuracy of 85% provided by the machine learning models exceeds the average clinical diagnosis accuracy of non-experts (73.8%) and average accuracy of movement disorder specialists (79.6% without follow-up, 83.9% after follow-up) with pathological post-mortem examination as ground truth.

2.3 Anitha , Nandhini, Sathish Raj, Nikitha , BOEARLY DETECTION OF PARKINSON'S DISEASE USING MACHINE LEARNING,2020

Abstract: Parkinson's disease (PD) is a neurodegenerative movement disease where the symptoms gradually develop starting with a slight tremor in one hand and a feeling of stiffness in the body and it becomes worse over time. It affects over 6 million people worldwide. At present there is no conclusive result for this disease by non-specialist clinicians, particularly in the early stage of the disease where identification of the symptoms are very difficult in its earlier stages. The proposed predictive analytics framework is a combination of K-means clustering and Decision Tree which is used to gain insights from patients. By using machine learning techniques, the problem can be solved with minimal error rate. Voice data sets obtained from the UCI Machine learning repository if given as the input for voice data analysis. Also our proposed system provides accurate results by integrating spiral drawing inputs of normal and Parkinson's affected patients. From these drawings Random forest classification algorithm is used which converts these drawings into pixels for classification and the extracted values are matched with the trained database to extract various features and results are produced with maximum accuracy. Also OpenCV (Open Source Computer Vision Library) a library of programming functions mainly aimed at real-time computer vision was built to provide an infrastructure for computer vision applications and to accelerate the use of machine perception in the real time. Thus our output will showcase the early detection of the disease and can be able to increase the lifespan of the diseased patient with proper treatments and medications leading to peaceful life.

2.4 Basil K Varghese, Geraldine Bessie Amali D*, Uma Devi K S,Prediction of Parkinson's Disease using Machine Learning Techniques on Speech dataset, 2019

Abstract:

In the present decade of accelerated advances in Medical Sciences, most studies fail to lay focus on aging diseases. These are diseases that display their symptoms at a much advanced stage and make a complete recovery almost improbable. Parkinson's disease (PD) is the second most commonly diagnosed neurodegenerative disorder of the brain. One could argue that it is almost incurable and inflicts a lot of pain on the patients. All these make it quite clear that there is an oncoming need for efficient, dependable and expandable diagnosis of Parkinson’s disease. A dilemma of this intensity requires the automating of the diagnosis to lead to accurate and reliable results. It has been observed that most PD Patients demonstrate some sort of impairment in speech or speech dysphonia, which makes speech measurements and indicators one of the most important aspects in prediction of PD. The aim of this work is to compare various machine learning models in the successful prediction of the severity of Parkinson’s disease and develop an effective and accurate model in order to help diagnose the disease accurately at an earlier stage which could in turn help the doctors to assist in the cure and recovery of PD Patients. For the aforementioned purpose we plan on using the Parkinson’s Tele monitoring dataset which was acquired from the UCIML repository.

2.5 John Michael Templeton, Christian Poellabauer & Sandra Schneider, Classification of Parkinson’s disease and its stages using machine learning, 2022

Abstract: As digital health technology becomes more pervasive, machine learning (ML) provides a robust way to analyze and interpret the myriad of collected features. The purpose of this preliminary work was to use ML classification to assess the benefits and relevance of neurocognitive features both tablet-based assessments and self-reported metrics, as they relate to Parkinson’s Disease (PD) and its stages [Hoehn and Yahr (H&Y) Stages 1–5]. Further, this work aims to compare perceived versus sensor-based neurocognitive abilities. In this study, 75 participants (n=50 PD; n=25 control) completed 14 tablet-based

neurocognitive functional tests (e.g., motor, memory, speech, executive, and multifunction), functional movement assessments (e.g., Berg Balance Scale), and standardized health questionnaires (e.g., PDQ-39). Decision tree classification of sensor-based features allowed for the discrimination of PD from healthy controls with an accuracy of 92.6%, and early and advanced stages of PD with an accuracy of 73.7%; compared to the current gold standard tools [e.g., standardized health questionnaires (78.3% accuracy) and functional movement assessments (70% accuracy)]. Significant features were also identified using decision tree classification. Device magnitude of acceleration was significant in 12 of 14 tests (85.7%), regardless of test type. For classification between diagnosed and control populations, 17 motor (e.g., device magnitude of acceleration), 9 accuracy (e.g., number of correct/incorrect interactions), and 8 timing features (e.g., time to between interactions) were significant. For classification between early (H&Y Stages 1 and 2) and advanced (H&Y Stages 3, 4, and 5) stages of PD, 7 motor, 12 accuracy, and 14 timing features were significant. Finally, this work depicts that perceived functionality of individuals with PD differed from sensor-based functionalities. In early-stage PD was shown to be 21.6% lower than sensor-based scores with notable perceived deficits in memory and executive function. However, individuals in advanced stages had elevated perceptions (1.57x) for executive and behavioral functions compared to early-stage populations. Machine learning in digital health systems allows for a more comprehensive understanding of neurodegenerative diseases and their stages and may also depict new features that influence the ways digital health technology should be configured.

2.6 Shrihari K Kulkarni1, K R Sumana2, Detection of Parkinson’s Disease Using Machine Learning and Deep Learning Algorithms, 2021

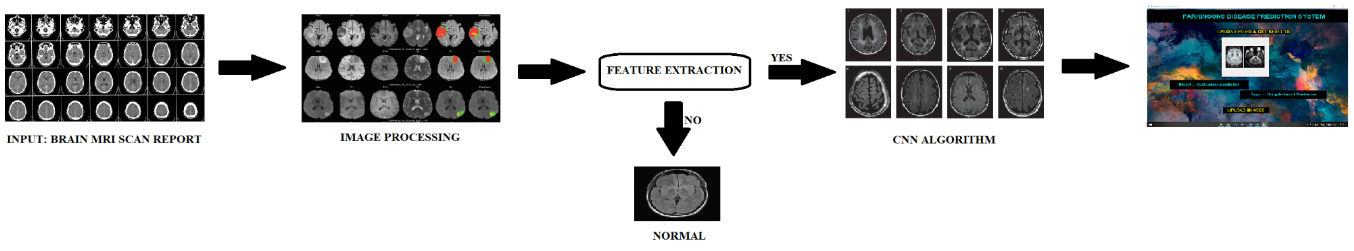
Abstract: Neurological diseases, like Parkinson's disease (PD), may be studied using biomarkers obtained from human speech. PD is a progressive neurodegenerative illness that affects around one million people. In the past, clinicians have relied on subjective grading systems to gauge the severity of Parkinson's disease. Difficulties with motor control make it possible to detect and diagnose PD via vocalization. Healthcare professionals could benefit from cheaper and more accurate diagnoses as a result of technological advancements and the widespread use of audio collecting devices in everyday life. We provide evidence to validate this concept here using a voice dataset collected from people with and without PD using Machine Learning algorithms: Decision Tree, Logistic Regression, and Naive Bayes and Deep Learning algorithm like Recurrent Neural Networks (RNN) by predicting with accuracy rate and performance comparison of all Machine Learning and Deep Learning algorithms.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE DIAGRAM

Infact, the adaptation process has to be automatic and dynamic to free the users with disabilities from the UI change control.



INPUT: BRAIN MRI SCAN REPORT
IMAGE PROCESSING
FEATURE EXTRACTION
YES
NO
NORMAL
CNN ALGORITHM
PARKINSONS DISEASE PREDICTION SYSTEM

FIG 3.1 Architecture diagram

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow; there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

3.2 USE-CASE DIAGRAM

A use case is a set of scenarios that describe an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. Here source is a user and destination is the receiver. The source sends the beacon signal to the receiver. Then the receiver sends the response to the source node.

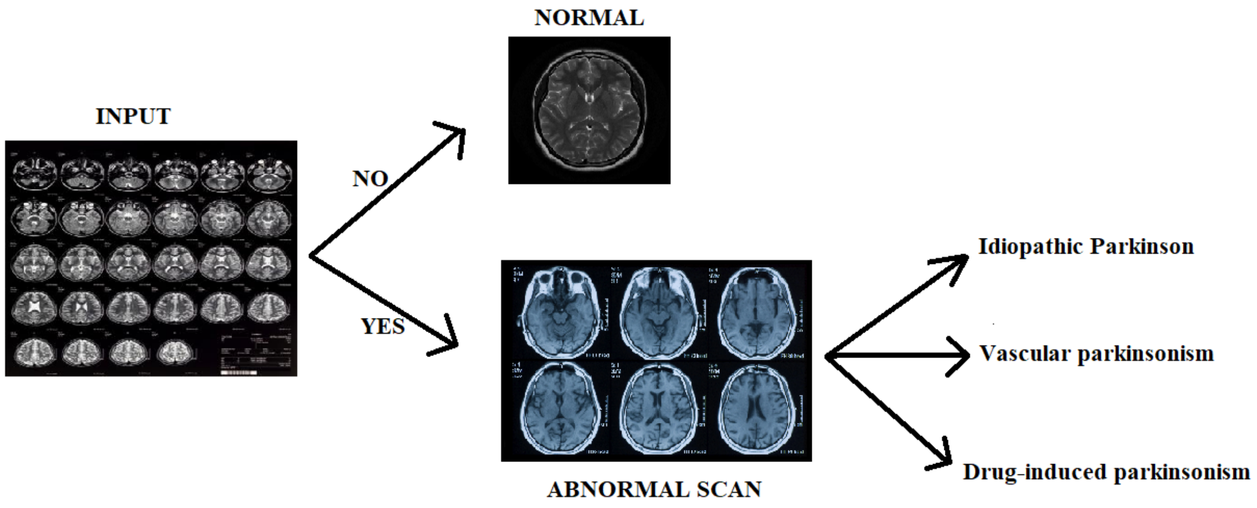


FIG 3.2 Use case diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

3.3 SEQUENCE DIAGRAM

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent.The consisting objects are Localizability testing,structure analysis,network adjustment and localizability aided localization.

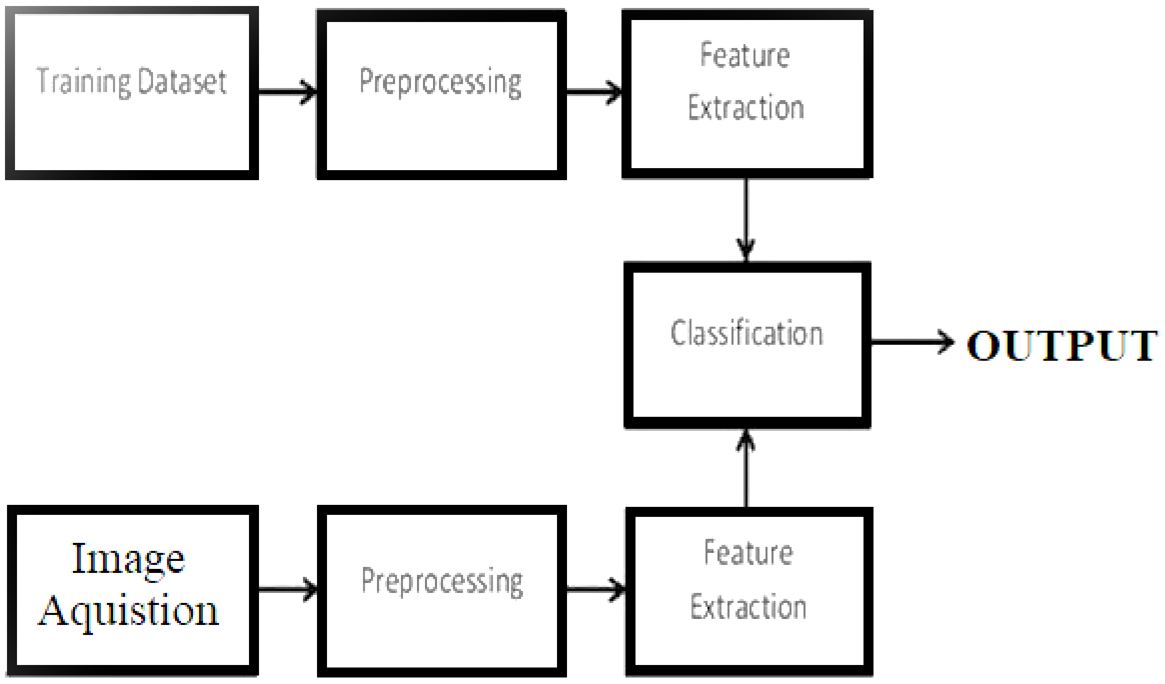


Fig 3.3 sequence diagram

A sequence diagram or system sequence diagram (SSD) shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

3.4 ARCHITECTURE FLOW DIAGRAM

The wireless Ad hoc and sensor networks to deploy the localization and non_localization nodes.In localizability aided localization would consists of localization testing,structure analysis,network adjustment by using tree prediction mobility.It is used to increase the mobility values.It is used to find the ranging measured.

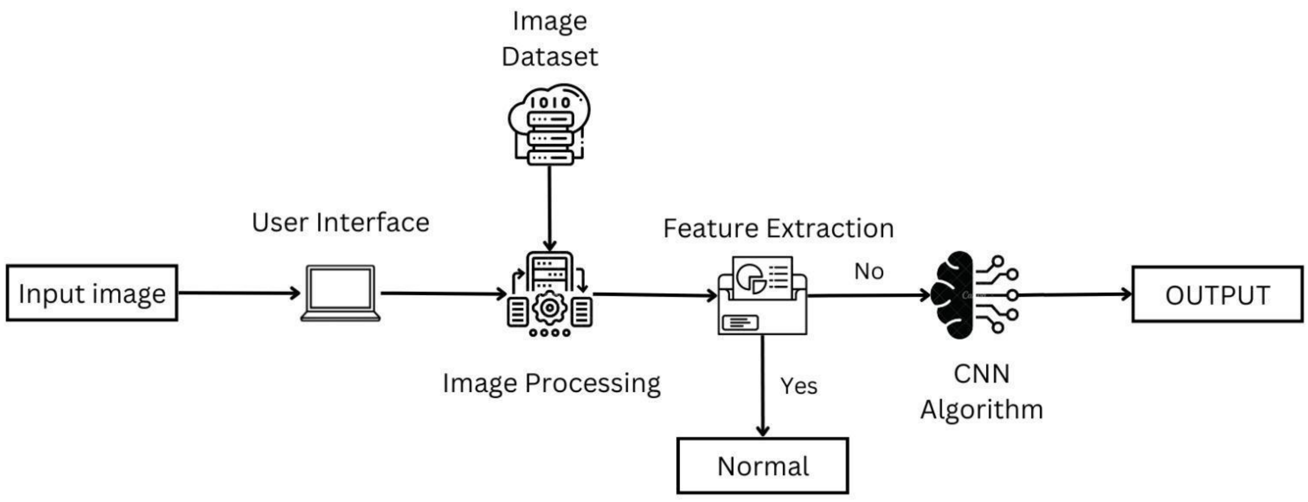
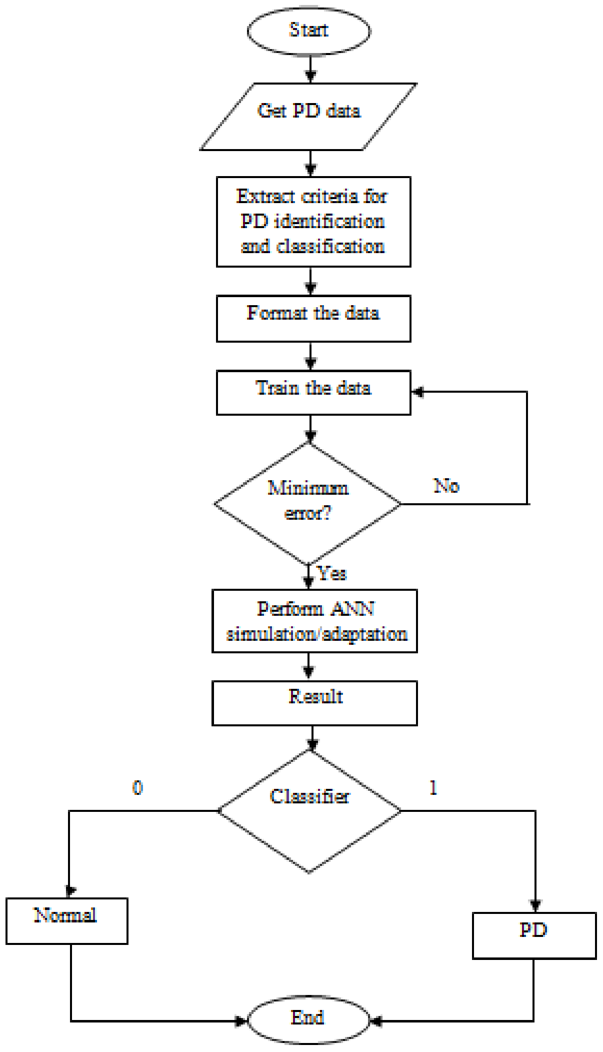


Fig 3.5 Architecture flow diagram

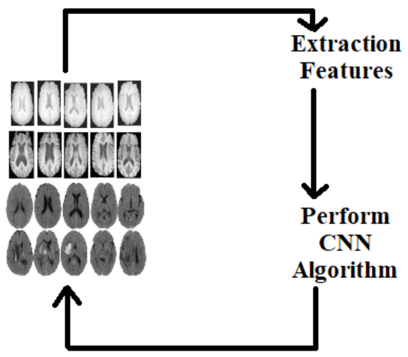
Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe the different perspectives when designing a system-conceptual, specification and implementation. Classes are composed of three things: name, attributes, and operations. Class diagrams also display relationships such as containment, inheritance, association etc. The association relationship is the most common relationship in a class diagram. The association shows the relationship between instances of classes.

3.5 DATA FLOW DIAGRAM

The data flow diagram is a graphic tool used for expressing system requirements in a graphical form. The data flow diagram would consist of step by step process; it contains source nodes ,channel splitting,Beacons used to receive the signal in narrow areas and to give acknowledgement to the sender.Localizability aided localization carried the process mobile networks behavior and to achieve the threshold.



3.6 BASIC FLOW:



3.7 PROPOSED ALGORITHM:

CNN ALGORITHM :

CNN :

Why CNN for Image Classification?

Image classification involves the extraction of features from the image to observe some patterns in the dataset. Using an ANN for the purpose of image classification would end up being very costly in terms of computation since the trainable parameters become extremely large.

For example, if we have a 50 X 50 image of a cat, and we want to train our traditional ANN on that image to classify it into a dog or a cat the trainable parameters become –

(50*50) * 100 image pixels multiplied by hidden layer + 100 bias + 2 * 100 output neurons + 2 bias = 2,50,30

Examples of different filters and their effects

Filters help us exploit the spatial locality of a particular image by enforcing a local connectivity pattern between neurons.

Convolution basically means a pointwise multiplication of two functions to produce

a third function. Here one function is our image pixels matrix and another is our filter. We slide the filter over the image and get the dot product of the two matrices. The resulting matrix is called an “Activation Map” or “Feature Map”.

Step 1: Choose a Dataset

Choose a dataset of your interest or you can also create your own image dataset for solving your own image classification problem. An easy place to choose a dataset is on kaggle.com.

The dataset I’m going with can be found here.

This dataset contains 12,500 augmented images of blood cells (JPEG) with accompanying cell type labels (CSV). There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil.

Here are all the libraries that we would require and the code for importing them.

Step 2: Prepare Dataset for Training

Preparing our dataset for training will involve assigning paths and creating categories(labels), resizing our images.

Resizing images into 200 X 200

Step 3: Create Training Data

Training is an array that will contain image pixel values and the index at which the image in the CATEGORIES list.

Step 4: Shuffle the Dataset

Step 5: Assigning Labels and Features

This shape of both the lists will be used in Classification using the NEURAL NETWORKS.

Step 6: Normalising X and converting labels to categorical data

Step 7: Split X and Y for use in CNN

Step 8: Define, compile and train the CNN Model

Step 9: Accuracy and Score of model

```
function XCOMPRESSCU(*pCurCU)
    M <= FastCUMope(PO,QP)
    if M 4 SPLIT then
        C2n <=CHECKINTRA(pCurCU)
    else
        C2n <= ∞
    end if

    if M != HOMO and Dcur < Dmax then

        Cn <= 0

        for i = 0 to 3 do

            pSubCUi <= pointer to SubCUi

             CN <= CN+ XCompressCU(pSubCUi)

        end for


    else

        CN <= ∞

    end if

    CHECKBESTMODE(C2N, CN)

    end function
```

Residual Networks (ResNet) in Keras

Very deep neural networks are hard to train as they are more prone to vanishing or exploding gradients. To solve this problem, the activation unit from a layer could be fed directly to a deeper layer of the network, which is termed as a skip connection.

This forms the basis of residual networks or ResNets. This post will introduce the basics the residual networks before implementing one in Keras.

Residual block

A building block of a ResNet is called a residual block or identity block. A residual block is simply when the activation of a layer is fast-forwarded to a deeper layer in the neural network.

As you can see in the image above, the activation from a previous layer is being added to the activation of a deeper layer in the network.

This simple tweak allows training much deeper neural networks.

In theory, the training error should monotonically decrease as more layers are added to a neural network. In practice however, for a traditional neural network, it will reach a point where the training error will start increasing.

ResNets do not suffer from this problem. The training error will keep decreasing as more layers are added to the network. In fact, ResNets have made it possible to train networks with more than 100 layers, even reaching 1000 layers.

Building a ResNet for image classification

Now, let’s build a ResNet with 50 layers for image classification using Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

In this case, we will use TensorFlow as the backend. Of course, feel free to grab the entire notebook and make all the necessary imports before starting.

Step 1: Define the identity block

Step 2: Convolution block

Step 3: Build the model

Step 4: Training

Step 5: Print the model summary

Algorithm 2. Pseudo code of the used preprocessing method.

Input: The raw 1D sensor signal (S) with size of 5625

Output: Graylevel image (Im) with size of 125 x 45

```
1: count = 1;
2: for i=1 to 125 do
3:     for j=1 to 45 do
4:         Im(i,j) = S(count);
5:         count = count + 1;
6:     end for j
7: end for i
8: Normalize Im by using min-max normalization.
```

GG16 implementation in Keras for beginners

VGG16 is a convolution neural net (CNN) architecture which was used to win ILSVR(Imagenet) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.

Here I first import all the libraries which i will need to implement VGG16. I will be using Sequential method as I am creating a sequential model. Sequential model means that all the layers of the model will be arranged in sequence. Here I have imported Image Data Generator from keras preprocessing. The objective of Image Data Generator is to import data with labels easily into the model. It is a very useful class as it has many function to rescale, rotate, zoom, flip etc. The most useful thing about this class is that it doesn’t affect the data stored on the disk. This class alters the data on the go while passing it to the model.

```
function CONV(i, [Bufz)
Co <= 0
while Co < Odo
    if Co, = 0 and i = 0 then
                c<=0
        READBIAS()
        READKERNEL(Co ~ Co + 31, KC)
    end if
    || PPMACConv(K., [Bufr)
    || PREFETCHKERNEL(C, + 32 ~ C, + 63, K.)
    C <= C
end while
end function
```

```
function CONV(TX, Ty, Ci)
TX <= 0, Ty <= 0,Ci <= 0
while Ty < Y do
    while Tx < X do
        while Ci< I do
            if Tx =0 and Ty = 0 then
                c<=0
                        READTILE(I  Buf., Tx, Ty, Ci)
            end if
            || Convop(Ci, [Buf.)
            || PREFETCHTILE(IBufc., Tx, Ty, Ci + 1
            C <= c
            end while
        end while
    end while
end function
```

MICE imputation

The fancyimpute package offers various robust machine learning models for imputing missing values. You can explore the complete list of imputers from the detailed documentation. Here, we will use IterativeImputer or popularly called MICE for imputing missing values.

The IterativeImputer performs multiple regressions on random samples of the data and aggregates for imputing the missing values. You will use the diabetes DataFrame for performing this imputation.

- Import IterativeImputer from fancyimpute.
- Copy diabetes to diabetes_mice_imputed.
- Create an IterativeImputer() object and assign it to mice_imputer.
- Impute the diabetes DataFrame.

CHAPTER 4

MODULES

4.1 MODULES LIST

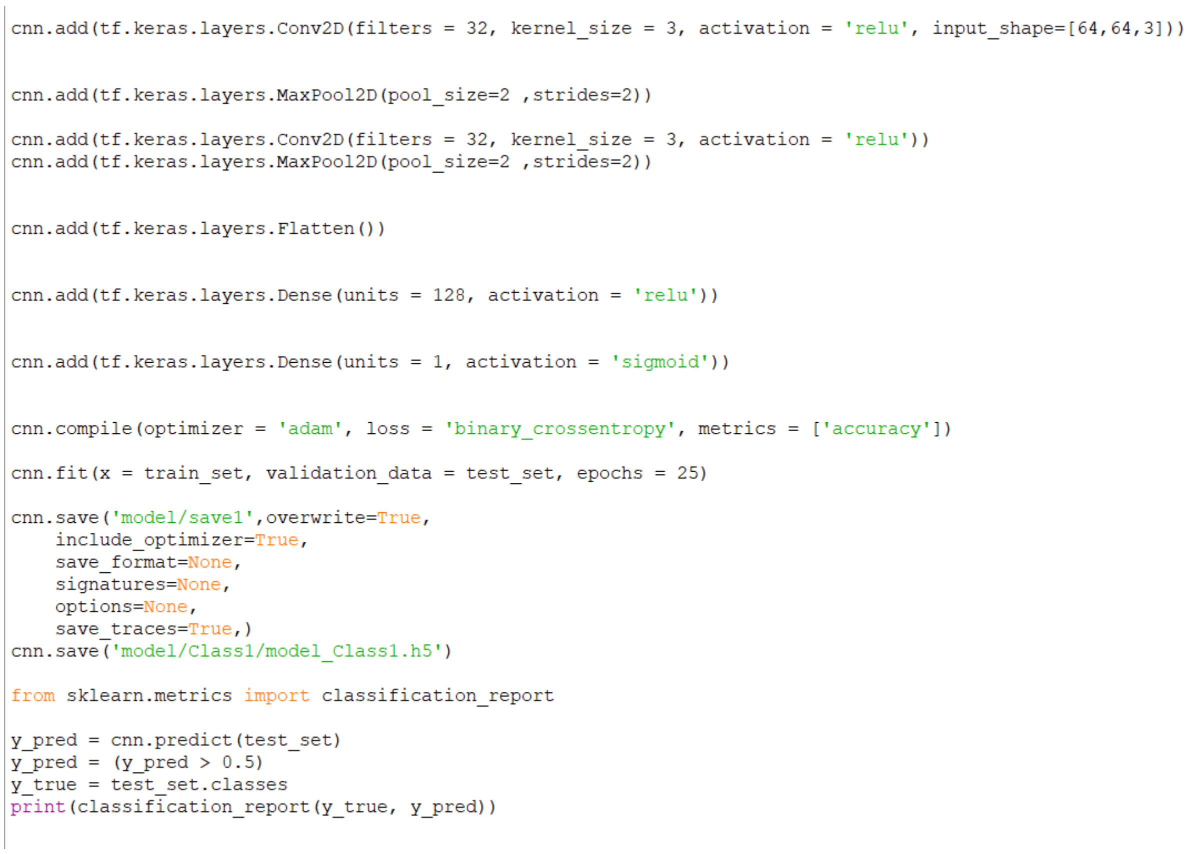
- Image Acquisition
- Image Pre-Processing
- Image Segmentation
- Feature Extraction
- MRI Classification
- Convolutional Neural Network

4.2 MODULES DESCRIPTION

4.2.1 CREATING CNN MODEL

1. Creating a data collection program to collect data from the user and detect the hands in the image frames, add crop the brain MRI scan.
2. Saving the collected image in a Data directory which contains the Train and Test folder.
3. Do this for all the dataset labels.
4. With the help of mediapipe module all the dataset of hand contain tracking pipeline which helps to increase the detection accuracy
5. Import the Tensorflow library to the python program for using the(.py)keras module to create CNN.
6. Use the Sequential module to build a CNN model and add the required layers such as Conv2D , Maxpool , Flatten , Dense layers.
7. The final result is shown in the GUI application.

```
import tensorflow as tf
from keras_preprocessing.image import ImageDataGenerator
from keras_preprocessing import image
import numpy as np
import easygui
import os
import serial
import absl.logging
absl.logging.set_verbosity(absl.logging.ERROR)


print(tf.__version__)


train_datagen = ImageDataGenerator(


        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

train_set = train_datagen.flow_from_directory(
        'Class1/training_set',

        target_size=(64, 64),

        batch_size=32,

        class_mode='binary')



test_datagen = ImageDataGenerator(rescale=1./255)

test_set = test_datagen.flow_from_directory(
        'Class1/test_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

print(test_set)
```

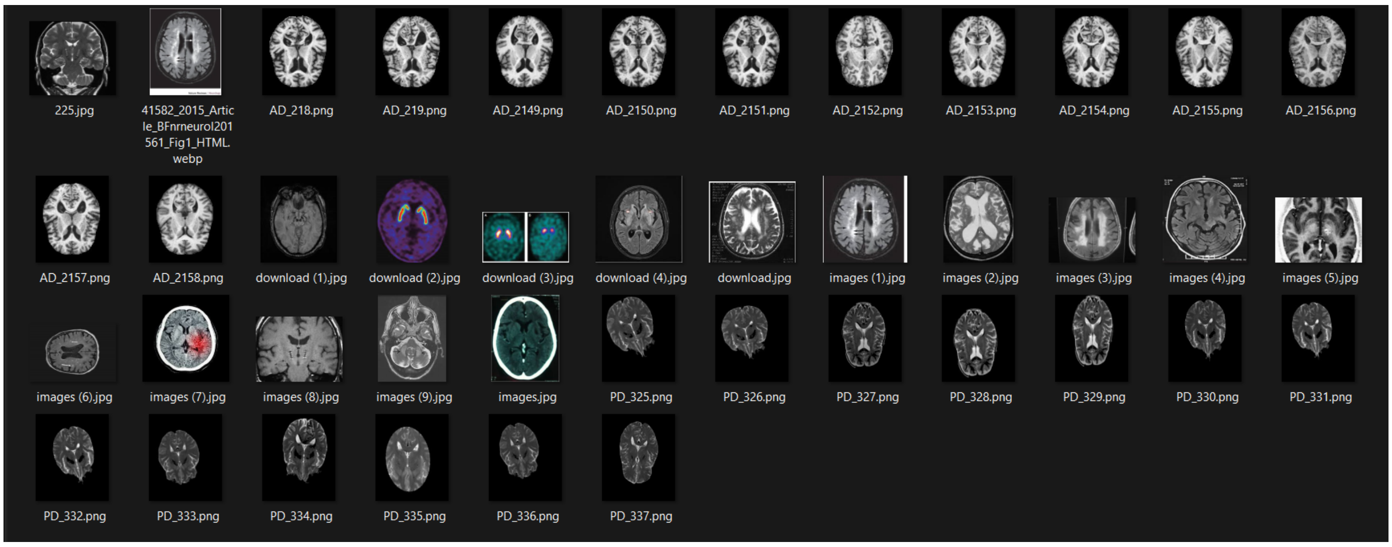


```
cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu', input_shape=[64,64,3]))


cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))

cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))


cnn.add(tf.keras.layers.Flatten())


cnn.add(tf.keras.layers.Dense(units = 128, activation = 'relu'))


cnn.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))


cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

cnn.fit(x = train_set, validation_data = test_set, epochs = 25)

cnn.save('model/save1',overwrite=True,
    include_optimizer=True,
    save_format=None,
    signatures=None,
    options=None,
    save_traces=True,)
cnn.save('model/Class1/model_Class1.h5')

from sklearn.metrics import classification_report

y_pred = cnn.predict(test_set)
y_pred = (y_pred > 0.5)
y_true = test_set.classes
print(classification_report(y_true, y_pred))
```

(Change)Fig 4.1 Train Model (sample code)

4.2.2 TESTING CNN MODEL

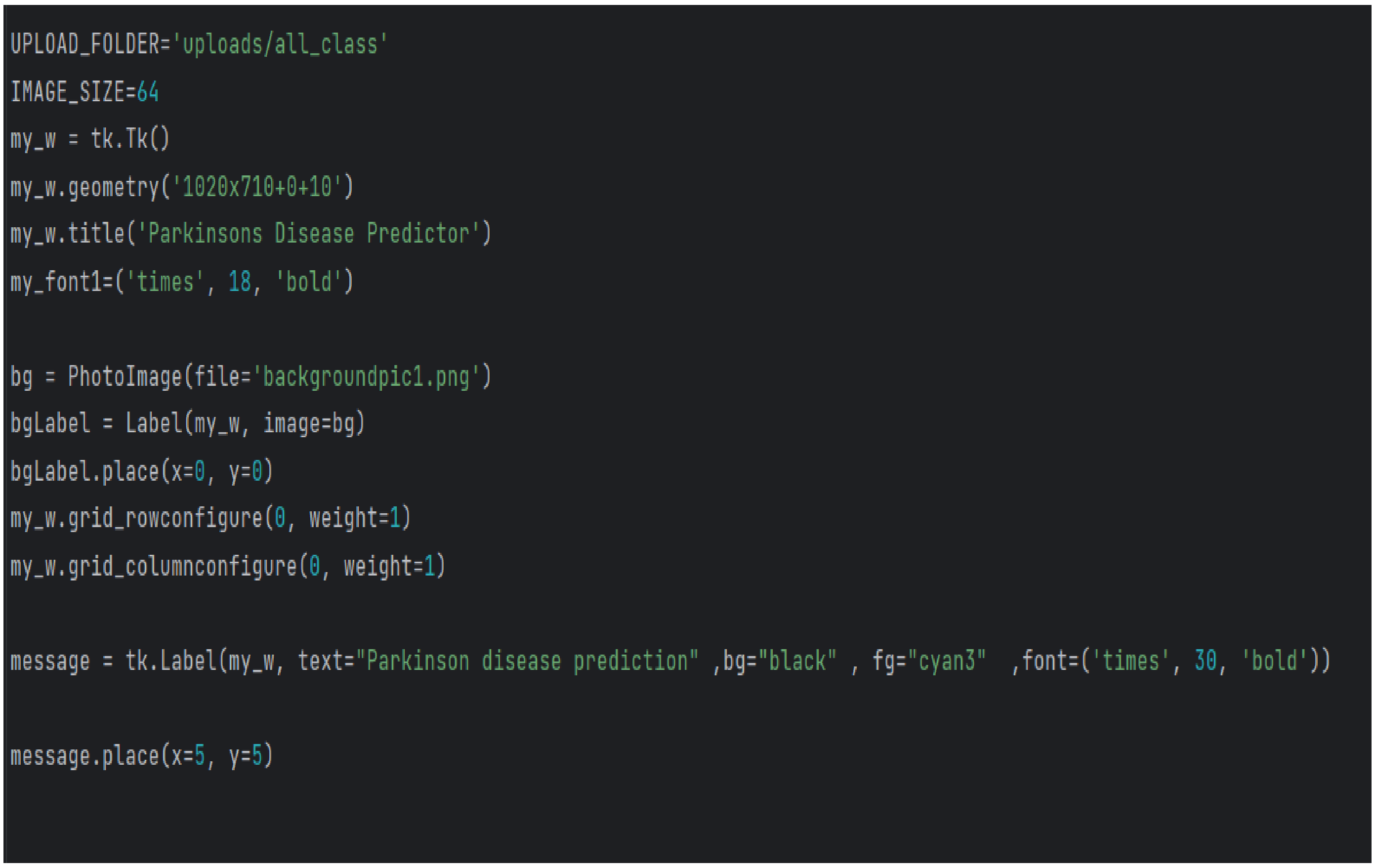
1. By using the YOLOv5 model we can predict the output of our project.
2. We use the Opencv and Pytorch library to predict the Hand Gestures and we also use this module to detect the hand in our image frame.
3. Using this, our image predicting model can detect the intended output with higher accuracy.
4. Our CNN Model is trained to localize the hand of the person who is trying to communicate, even in a messy background.
5. It is trained to get the most desired aspect to detect the hand tracking with higher precision.
6. The '0' in the VideoCapture() method in the below program denotes the external camera connected to the device.
7. The maxHands=1 in the HandDetector() method in the above program denotes that we are detecting just the one hand of the person/user



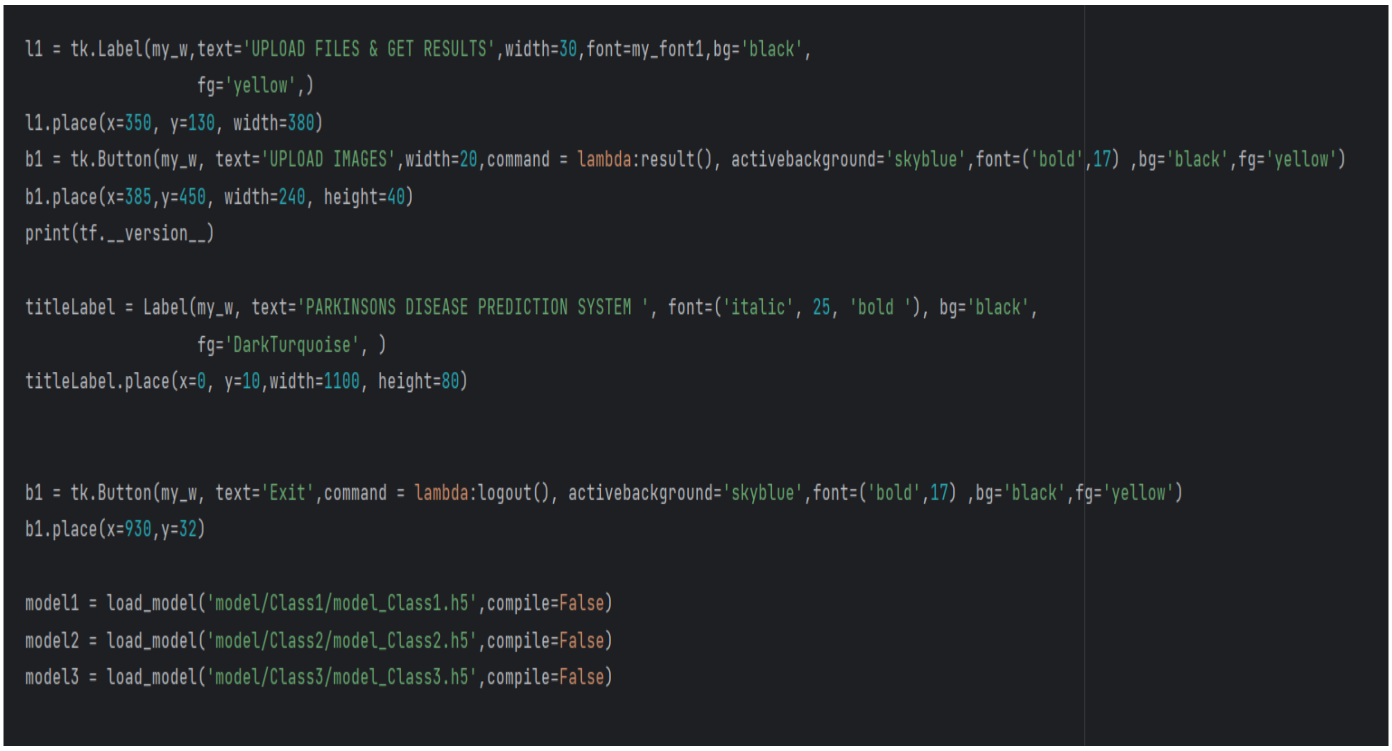
(Change)Fig 4.2 Dataset Collection

4.2.3 TKINTER GUI APPLICATION

1. Create a gui application. Command Line: from flask import tkinter, PIL, keras, easy gui.
2. Adding required functions to render the templates.
3. Imported the prediction program to flask app import opencv.
4. Imported train list to the flask app. import train list
5. Adding required functions to render the templates.
6. Run the app.py (Flask App).



```
UPLOAD_FOLDER='uploads/all_class'
IMAGE_SIZE=64
my_w = tk.Tk()
my_w.geometry('1020x710+0+10')
my_w.title('Parkinsons Disease Predictor')
my_font1=('times', 18, 'bold')

bg = PhotoImage(file='backgroundpic1.png')
bgLabel = Label(my_w, image=bg)
bgLabel.place(x=0, y=0)
my_w.grid_rowconfigure(0, weight=1)
my_w.grid_columnconfigure(0, weight=1)

message = tk.Label(my_w, text="Parkinson disease prediction" ,bg="black" , fg="cyan3"  ,font=('times', 30, 'bold'))

message.place(x=5, y=5)
```



```
l1 = tk.Label(my_w,text='UPLOAD FILES & GET RESULTS',width=30,font=my_font1,bg='black',
                fg='yellow',)
l1.place(x=350, y=130, width=380)
b1 = tk.Button(my_w, text='UPLOAD IMAGES',width=20,command = lambda:result(), activebackground='skyblue',font=('bold',17) ,bg='black',fg='yellow')
b1.place(x=385,y=450, width=240, height=40)
print(tf.__version__)

titleLabel = Label(my_w, text='PARKINSONS DISEASE PREDICTION SYSTEM ', font=('italic', 25, 'bold '), bg='black',
                fg='DarkTurquoise', )
titleLabel.place(x=0, y=10,width=1100, height=80)


b1 = tk.Button(my_w, text='Exit',command = lambda:logout(), activebackground='skyblue',font=('bold',17) ,bg='black',fg='yellow')
b1.place(x=930,y=32)

model1 = load_model('model/Class1/model_Class1.h5',compile=False)
model2 = load_model('model/Class2/model_Class2.h5',compile=False)
model3 = load_model('model/Class3/model_Class3.h5',compile=False)
```

(Change) Fig 4.3 Sample Code for data collection

CHAPTER 5

SYSTEM SPECIFICATION

5.1 SOFTWARE REQUIREMENT SPECIFICATION

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description as functional representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

5.2 SYSTEM REQUIREMENTS

5.2.1 HARDWARE REQUIREMENT

Operating System	**:**	**Windows,Linux,MacOS**
Processor	**:**	**Intel Core 2 DUO / AMD Athlon 64 X2 5600+**
Video Card	**:**	**NVIDIA Ge Force 7600 GT / ATI Radeon HD 2600 XT / Intel HD Graphics 3000**
Memory	**:**	**8 GB RAM**
Resolution	**:**	**1024*768 minimum display resolution**

5.2.2 SOFTWARE REQUIREMENTS:

Software Tool	**:**	**Netbeans IDE 7.3.1,MySQL Query Browser**
Operating System	**:**	**Windows, Linux, MacOS**
Processors	**:**	**Any Intel or AMD X86-64 processor**
RAM	**:**	**8 GB**
Graphics Card	**:**	**No Specific graphics card required**

CHAPTER 6

SOFTWARE DESCRIPTION

6.1 SOFTWARE REQUIREMENT SPECIFICATION

6.1.1 INTRODUCTION TO PYTHON

Python is a high-level object-oriented programming language that was created by Guido van Rossum. It is also called general-purpose programming language as it is used in almost every domain we can think of as mentioned below:

- Web Development
- Software Development
- Game Development
- AI & ML
- Data Analytics

This list can go on as we go but why python is so much popular let's see it in the next topic.

Every Programming language serves some purpose or use-case according to a domain. For example, Javascript is the most popular language amongst web developers as it gives the developer the power to handle applications via different frameworks like react, vue, angular which are used to build beautiful User Interfaces. Similarly, they have pros and cons at the same time. So if we consider python it is general-purpose which means it is widely used in every domain. The reason is it's very simple to understand, scalable because the speed of development is so fast. Now you get the idea why besides learning python it doesn't require any programming background so that's why it's popular amongst developers as well. Python has simpler syntax similar to the English language and also the syntax allows developers to write programs with fewer lines of code. Since it is open-source there are many libraries available that make developers' jobs easy, ultimately resulting in high productivity. They can easily focus on business logic and Its demanding skills in the digital era where information is available in large data sets.

6.1.2 SQLite:

SQLite is a database engine written in the C programming language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems. Many programming languages have bindings to the SQLite library. It generally follows PostgreSQL syntax, but does not enforce type checking by default. This means that one can, for example, insert a string into a column defined as an integer.

SQLite is a lightweight, serverless, and self-contained database engine that is commonly used in embedded systems and small-scale applications. It provides a simple and efficient way to store and retrieve data. In the code, SQLite is utilized for user registration and login functionality. The database file "db_member.db" is created if it doesn't exist and a table named "member" is created to store user information such as username, password, firstname, and lastname.

6.1.3 Tensorflow

TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive set of tools and functionalities for building and training machine learning models. TensorFlow offers a flexible architecture that allows users to define and execute computational graphs, which represent mathematical operations and their dependencies.

Key features of TensorFlow include:

1. Automatic differentiation: TensorFlow automatically calculates gradients, which are essential for optimizing models during the training process.

2. GPU acceleration: TensorFlow supports GPU acceleration, enabling faster computations for training large-scale models.

3. Model deployment: TensorFlow allows models to be deployed in a variety of environments, including desktops, servers, mobile devices, and even web browsers.

4. Community and ecosystem: TensorFlow has a large and active community, providing extensive documentation, tutorials, and support. It also has a wide range of pre-trained models and tools for various machine learning tasks.

6.1.4 Keras:

Keras is a high-level neural networks API written in Python. It is built on top of TensorFlow (as well as other backend engines like Theano and Microsoft Cognitive Toolkit) and provides a user-friendly interface for designing and training deep learning models. Keras abstracts away many low-level implementation details and focuses on ease of use, making it suitable for both beginners and experienced deep learning practitioners.

Key features of Keras include:

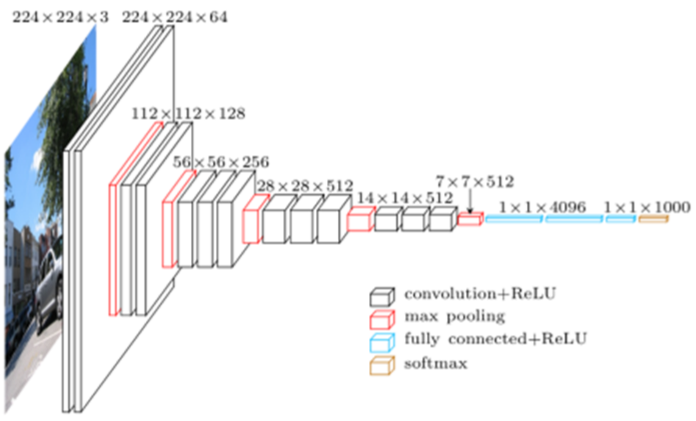
1. Simple and intuitive API: Keras provides a user-friendly and intuitive API for defining and training neural network models. It offers a high-level abstraction that allows users to easily build complex network architectures.

2. Modularity and extensibility: Keras promotes modular model design, allowing users to easily stack layers, connect different components, and build custom models. It also supports the creation of custom layers and loss functions.

3. Model interoperability: Keras seamlessly integrates with other deep learning libraries such as TensorFlow. Users can use TensorFlow as the backend while leveraging the simplicity and flexibility of Keras.

4. Broad applicability: Keras supports a wide range of applications, including image classification, natural language processing, sequence modeling, and more. It provides pre-built models and utilities for common tasks, making it convenient for rapid prototyping.

By combining TensorFlow and Keras, developers can leverage the power of TensorFlow's computation graph and low-level operations, while benefiting from Keras' simplicity and high-level abstractions for building and training neural networks.


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

In conclusion, the integration of artificial intelligence techniques, such as machine learning and deep learning, has enabled the development of adaptive user interfaces that can empower users with disabilities, particularly those with hearing impairments. Our proposed project leverages the power of TensorFlow and Mediapipe to create a feasible communication channel between hearing-impaired and normal individuals. With an accuracy of 90.2% under good lighting conditions, we believe that our approach has the potential to revolutionize the way hearing-impaired individuals interact with the world. As a next step, we plan to extend our system to real-time data to make it more practical and useful in real-life scenarios. This project is just the beginning of what could be a transformative technology for the disabled community, and we are excited to see where it leads in the future.

7.2 FUTURE ENHANCEMENT

Although this study focused on five other convolutional models and transfer learning designs for Parkinson's disease in the medical imaging field, further research is needed. We will investigate more significant and influential deep CNN models for Parkinson disease classification and conduct segmentation with reduced time complexity in future approaches. Also, to improve the accuracy of the proposed model, we will increase the number of MRI scans in the dataset used for this study. Furthermore, we will also be applying the proposed approach to other medical images such as x-ray, computed tomography (CT), and ultrasound which may serve as a foundation for future research.

SCREENSHOT 1

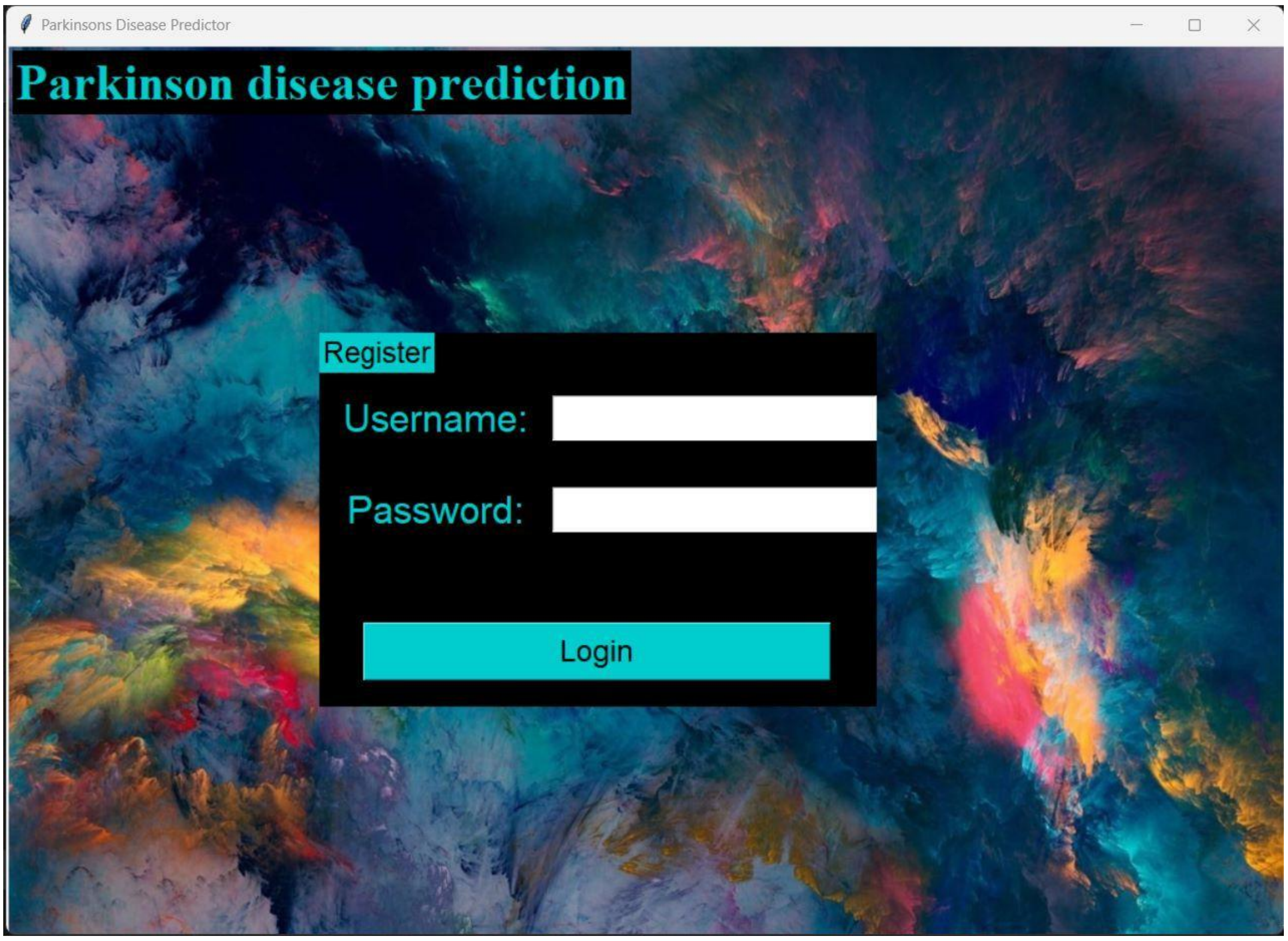


Fig 1. Login Page

In this page, the user can log in directly if the user has credentials else the user can register about and get credentials

SCREENSHOT 2

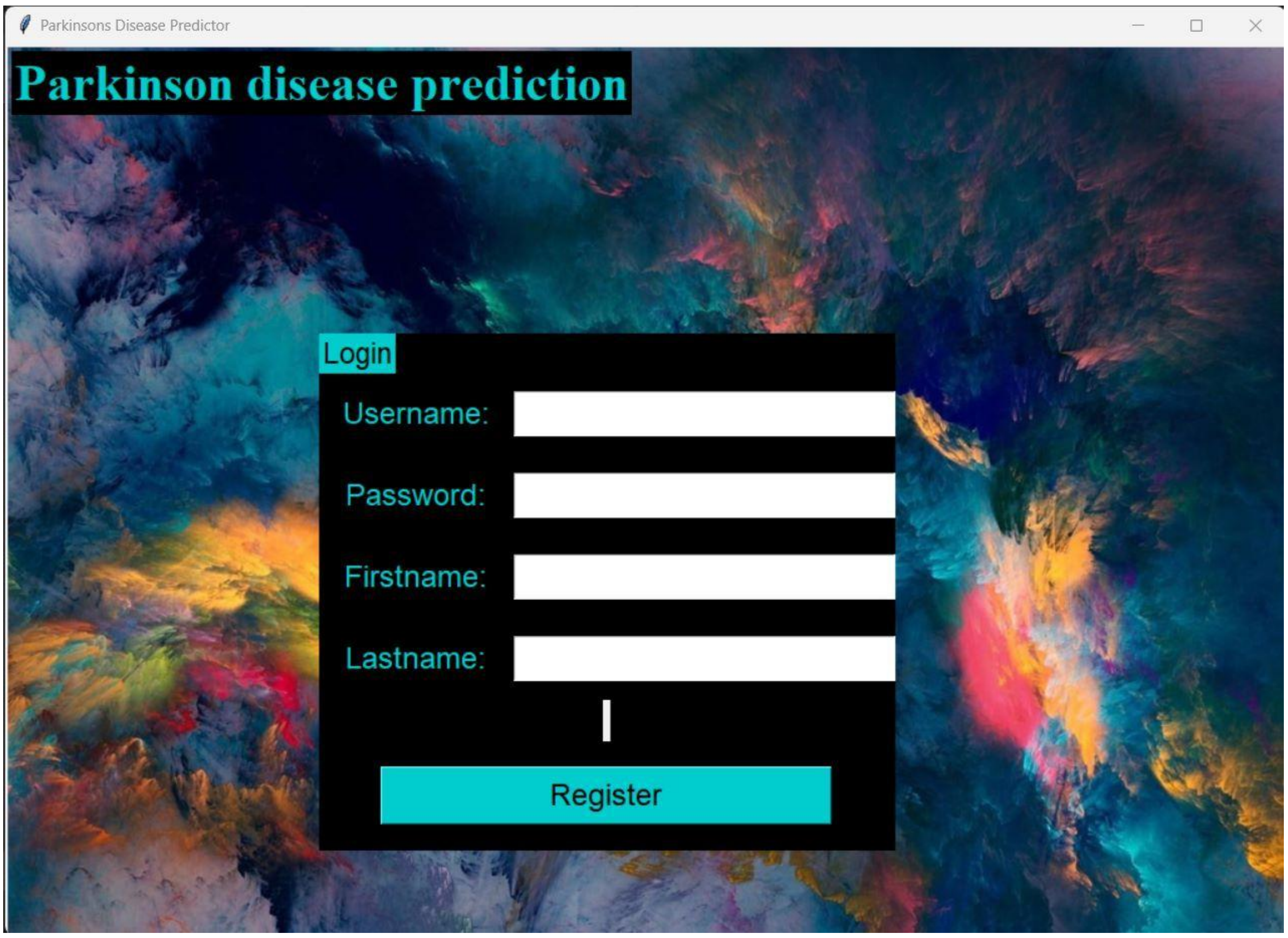


Fig 2. Register page

SCREENSHOT 3

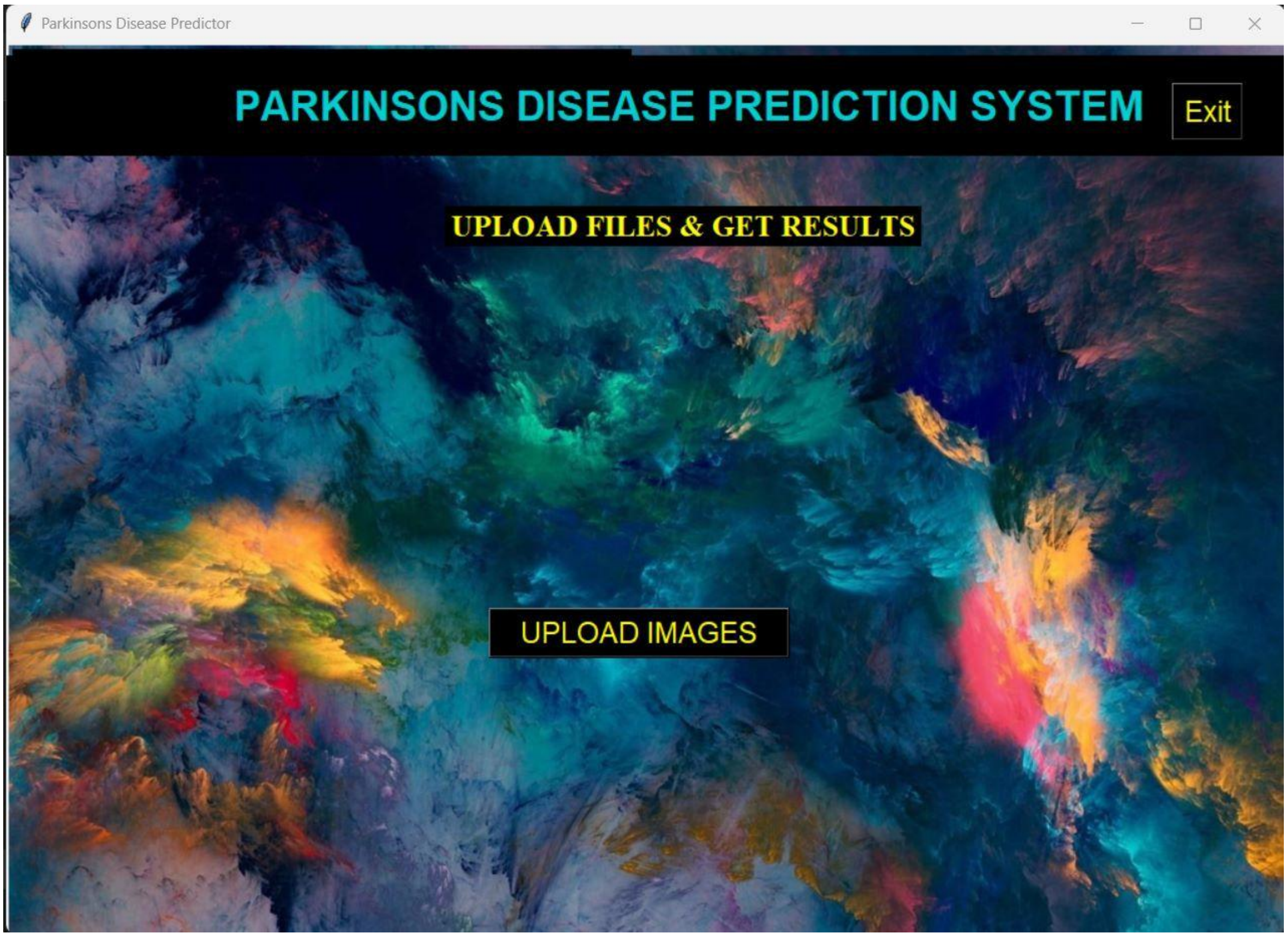
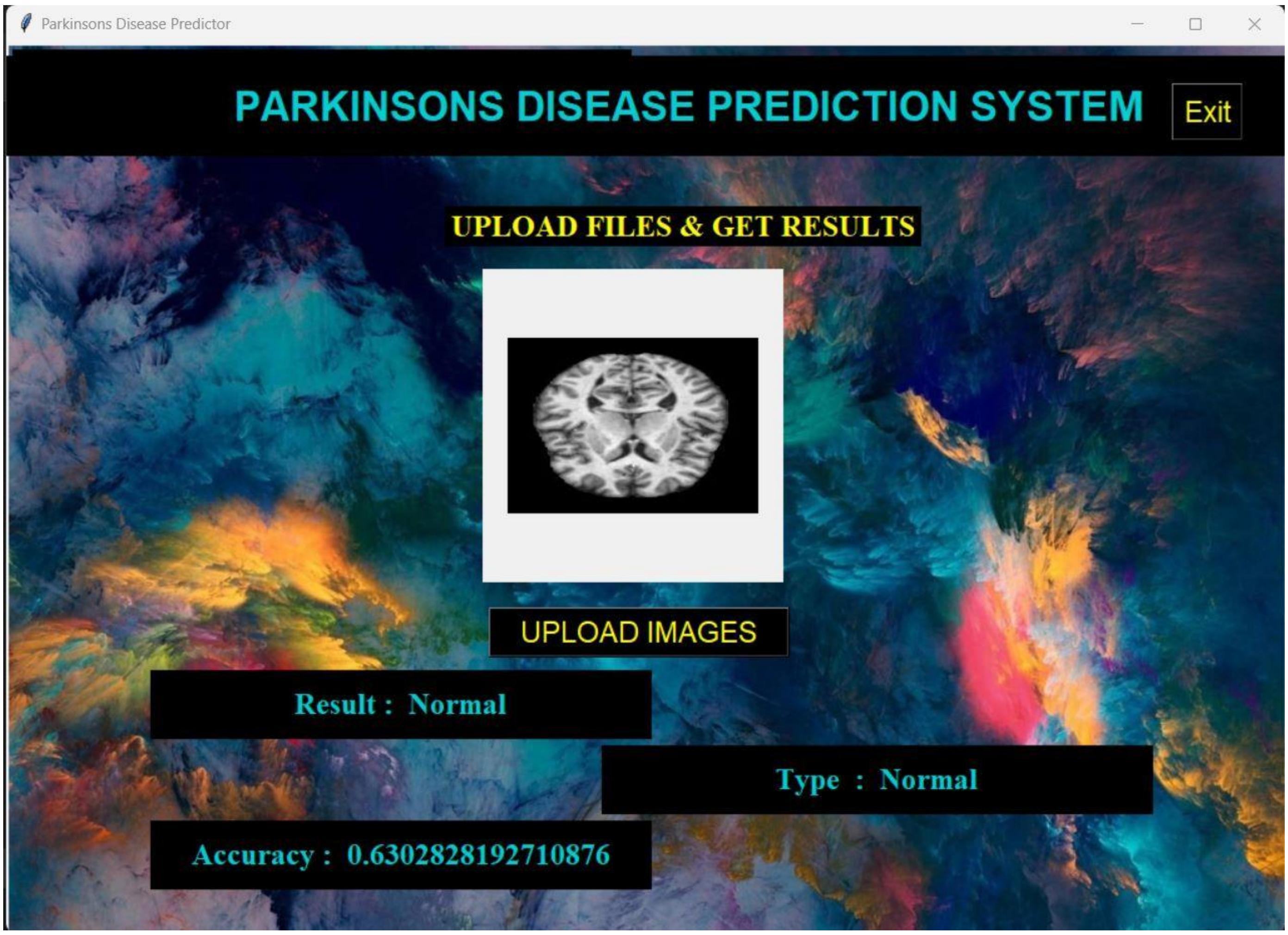

Parkinsons Disease Predictor
PARKINSONS DISEASE PREDICTION SYSTEM
Exit
UPLOAD FILES & GET RESULTS
UPLOAD IMAGES

Fig 3. Image uploading page

Here, the user can upload the MRI scan of the patient.

SCREENSHOT 4


Parkinsons Disease Predictor
PARKINSONS DISEASE PREDICTION SYSTEM
Exit
UPLOAD FILES & GET RESULTS
UPLOAD IMAGES

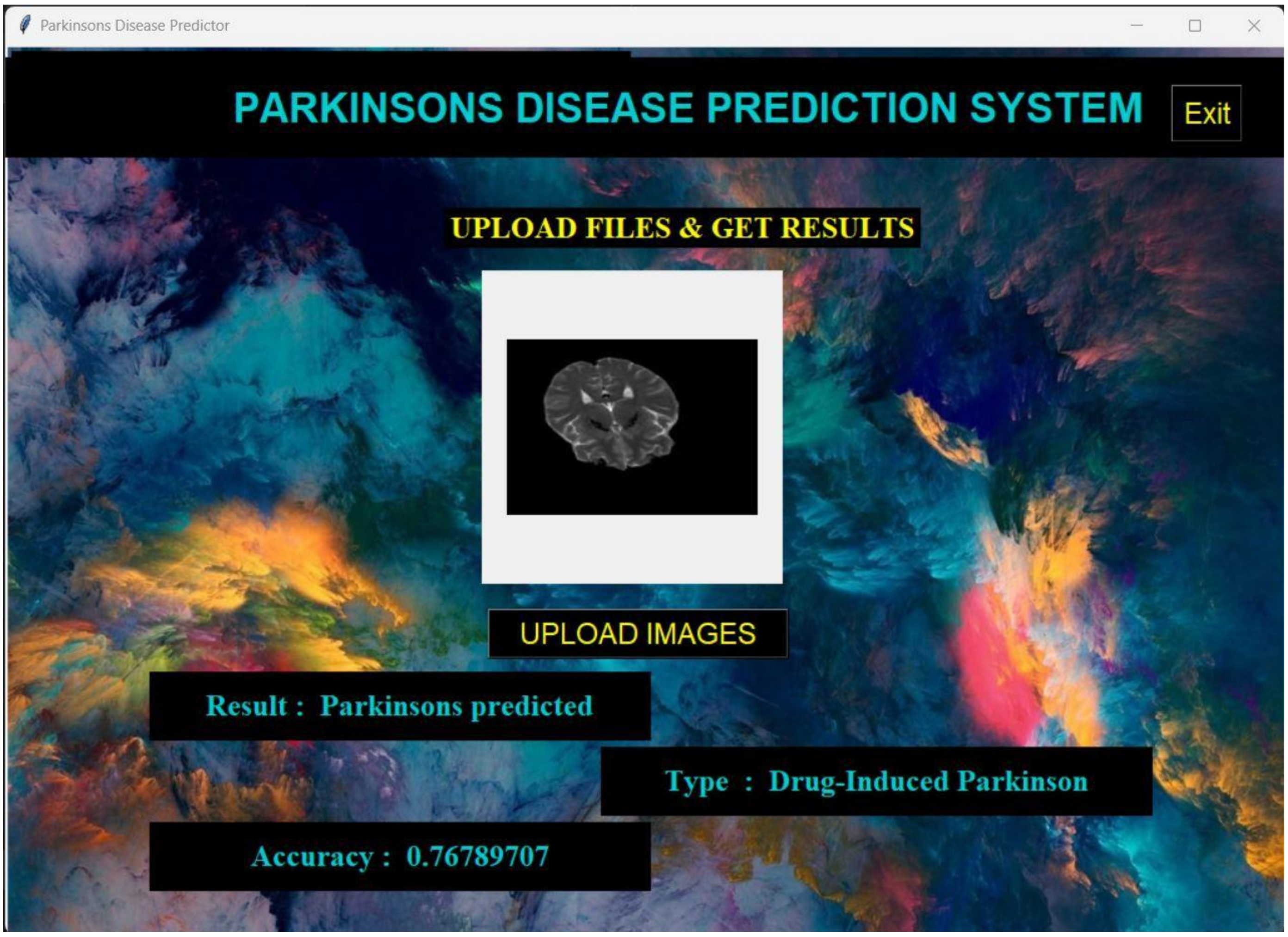
Result : Normal

Type : Normal

Accuracy : 0.6302828192710876

Fig 4. Result Page 1

SCREENSHOT 5


Parkinsons Disease Predictor
PARKINSONS DISEASE PREDICTION SYSTEM
Exit
UPLOAD FILES & GET RESULTS
UPLOAD IMAGES

Result : Parkinsons predicted

Type : Drug-Induced Parkinson

Accuracy : 0.76789707

Fig 5. Result Page 2

IMPLEMENTATION CODE :

class1.py

```
import tensorflow as tf
from keras_preprocessing.image import ImageDataGenerator
from keras_preprocessing import image
import numpy as np
import easygui
import os
import serial
import absl.logging
absl.logging.set_verbosity(absl.logging.ERROR)


print(tf.__version__)


train_datagen = ImageDataGenerator(


        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

train_set = train_datagen.flow_from_directory(
        'Class1/training_set',

        target_size=(64, 64),

        batch_size=32,

        class_mode='binary')



test_datagen = ImageDataGenerator(rescale=1./255)

test_set = test_datagen.flow_from_directory(
        'Class1/test_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

print(test_set)


cnn = tf.keras.models.Sequential()
cnn = tf.keras.models.Sequential()


cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu', input_shape=[64,64,3]))


cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))

cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))


cnn.add(tf.keras.layers.Flatten())


cnn.add(tf.keras.layers.Dense(units = 128, activation = 'relu'))


cnn.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))


cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

cnn.fit(x = train_set, validation_data = test_set, epochs = 25)

cnn.save('model/save1',overwrite=True,
    include_optimizer=True,
    save_format=None,
    signatures=None,
    options=None,
    save_traces=True,)
cnn.save('model/Class1/model_Class1.h5')

from sklearn.metrics import classification_report

y_pred = cnn.predict(test_set)
y_pred = (y_pred > 0.5)
y_true = test_set.classes
print(classification_report(y_true, y_pred))
```

class2.py

```
import tensorflow as tf

from keras_preprocessing.image import ImageDataGenerator
from keras_preprocessing import image
import numpy as np
import easygui
import os
import serial
import absl.logging
absl.logging.set_verbosity(absl.logging.ERROR)
print(tf.__version__)


train_datagen = ImageDataGenerator(


        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

train_set = train_datagen.flow_from_directory(
        'Class2/training_set',

        target_size=(64, 64),

        batch_size=32,

        class_mode='binary')



test_datagen = ImageDataGenerator(rescale=1./255)

test_set = test_datagen.flow_from_directory(
        'Class2/test_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

print(test_set)


cnn = tf.keras.models.Sequential()
```

```
cnn.add(tf.keras.layers.Flatten())


cnn.add(tf.keras.layers.Dense(units = 128, activation = 'relu'))


cnn.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))


cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

cnn.fit(x = train_set, validation_data = test_set, epochs = 25)
cnn.save('model/save1',overwrite=True,
    include_optimizer=True,
    save_format=None,
    signatures=None,
    options=None,
    save_traces=True,)
cnn.save('model/Class2/model_Class2.h5')

from sklearn.metrics import classification_report

y_pred = cnn.predict(test_set)
y_pred = (y_pred > 0.5)
y_true = test_set.classes
print(classification_report(y_true, y_pred))

a="continue"
while a=="continue":
   image11 = easygui.fileopenbox()
   test_image2 = image.load_img(image11, target_size = (64, 64))
   test_image2 = image.img_to_array(test_image2)
   test_image2 = np.expand_dims(test_image2, axis = 0)

   result2 = cnn.predict(test_image2)
   print(result2)
   if result2[0][0] == 0:
      prediction2 = 'Idiopathic Parkinson'
   else:
      prediction2 = 'other Category'


   print(prediction2)
   print("Type your ans")
   b=str(input("continue or  exit ; "))
   a=b.lower()
```

class3.py

```
import tensorflow as tf
from keras_preprocessing.image import ImageDataGenerator
from keras_preprocessing import image
import numpy as np
import easygui
import os
import serial
import absl.logging
absl.logging.set_verbosity(absl.logging.ERROR)
print(tf.__version__)


train_datagen = ImageDataGenerator(


        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

train_set = train_datagen.flow_from_directory(
        'Class3/training_set',

        target_size=(64, 64),

        batch_size=32,

        class_mode='binary')



test_datagen = ImageDataGenerator(rescale=1./255)

test_set = test_datagen.flow_from_directory(
        'Class3/test_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

print(test_set)


cnn = tf.keras.models.Sequential()
```

```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))

cnn.add(tf.keras.layers.Conv2D(filters = 32, kernel_size = 3, activation = 'relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 ,strides=2))


cnn.add(tf.keras.layers.Flatten())


cnn.add(tf.keras.layers.Dense(units = 128, activation = 'relu'))


cnn.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))


cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

cnn.fit(x = train_set, validation_data = test_set, epochs = 25)
cnn.save('model/save1',overwrite=True,
    include_optimizer=True,
    save_format=None,
    signatures=None,
    options=None,
    save_traces=True,)
cnn.save('model/Class3/model_Class3.h5')

a="continue"
while a=="continue":
   image11 = easygui.fileopenbox()
   test_image2 = image.load_img(image11, target_size = (64, 64))
   test_image2 = image.img_to_array(test_image2)
   test_image2 = np.expand_dims(test_image2, axis = 0)

   result2 = cnn.predict(test_image2)
   print(result2)
   if result2[0][0] == 1:
      prediction2 = 'Drug-induced parkinsons '
   else:
      prediction2 = 'Vascular parkinsonism'


   print(prediction2)
   print("Type your ans")
   b=str(input("continue or  exit ; "))
   a=b.lower()
```

main.py

```
import tensorflow as tf
import glob
import random
from keras_preprocessing.image import ImageDataGenerator
from keras_preprocessing import image
import numpy as np
import shutil
import easygui
from keras.models import load_model
import os
import tkinter as tk
from tkinter import *
from tkinter import filedialog
from tkinter.filedialog import askopenfile
from PIL import Image, ImageTk
import sqlite3


UPLOAD_FOLDER='uploads/all_class'
IMAGE_SIZE=64
my_w = tk.Tk()
my_w.geometry('1020x710+0+10')
my_w.title('Parkinsons Disease Predictor')
my_font1=('times', 18, 'bold')

bg = PhotoImage(file='backgroundpic1.png')
bgLabel = Label(my_w, image=bg)
bgLabel.place(x=0, y=0)
my_w.grid_rowconfigure(0, weight=1)
my_w.grid_columnconfigure(0, weight=1)

message = tk.Label(my_w, text="Parkinson disease prediction" ,bg="black" , fg="cyan3"  ,font=('times', 30, 'bold'))

message.place(x=5, y=5)


#========================================VARIABLES=====================================
USERNAME = StringVar()
PASSWORD = StringVar()
FIRSTNAME = StringVar()
LASTNAME = StringVar()



#=======================================METHODS======================================

def Database():
    global conn, cursor
    conn = sqlite3.connect("db_member.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, username TEXT, password TEXT, firstname TEXT, lastname TEXT)")


def Exit():
    result = tkMessageBox.askquestion('System', 'Are you sure you want to exit?', icon="warning")
    if result == 'yes':
        my_w.destroy()
        exit()



def LoginForm():
    global LoginFrame, lbl_result1
    LoginFrame = Frame(my_w,bg="black")
    LoginFrame.place(x=250,y=230)
    #LoginFrame.pack(side=TOP, pady=100)
    lbl_username = Label(LoginFrame, text="Username:", font=('arial', 23), bd=18,bg="black",fg="cyan3")
    lbl_username.grid(row=1)
    lbl_password = Label(LoginFrame, text="Password:", font=('arial', 23), bd=18,bg="black",fg="cyan3")
    lbl_password.grid(row=2)
    lbl_result1 = Label(LoginFrame, text="", font=('arial', 18),bg="black",fg="cyan3")
    lbl_result1.grid(row=3, columnspan=2)
    username = Entry(LoginFrame, font=('arial', 20), textvariable=USERNAME, width=17)
    username.grid(row=1, column=1)
    password = Entry(LoginFrame, font=('arial', 20), textvariable=PASSWORD, width=17, show="*")
    password.grid(row=2, column=1)
    btn_login = Button(LoginFrame, text="Login", font=('arial', 18), width=26, command=Login,bg="cyan3",fg="black")
    btn_login.grid(row=4, columnspan=2, pady=20)
    lbl_register = Label(LoginFrame, text="Register", font=('arial', 17),bg="cyan3",fg="black")
    lbl_register.grid(row=0, sticky=W)
    lbl_register.bind('<Button-1>', ToggleToRegister)
```

```python
def RegisterForm():
    global RegisterFrame, lbl_result2
    RegisterFrame = Frame(my_w,bg="black")
    RegisterFrame.place(x=250,y=230)
    #RegisterFrame.pack(side=TOP, pady=40)
    lbl_username = Label(RegisterFrame, text="Username:", font=('arial', 18), bd=18,bg="black",fg="cyan3")
    lbl_username.grid(row=1)
    lbl_password = Label(RegisterFrame, text="Password:", font=('arial', 18), bd=18,bg="black",fg="cyan3")
    lbl_password.grid(row=2)
    lbl_firstname = Label(RegisterFrame, text="Firstname:", font=('arial', 18), bd=18,bg="black",fg="cyan3")
    lbl_firstname.grid(row=3)
    lbl_lastname = Label(RegisterFrame, text="Lastname:", font=('arial', 18), bd=18,bg="black",fg="cyan3")
    lbl_lastname.grid(row=4)
    lbl_result2 = Label(RegisterFrame, text="", font=('arial', 18))
    lbl_result2.grid(row=5, columnspan=2)
    username = Entry(RegisterFrame, font=('arial', 20), textvariable=USERNAME, width=20)
    username.grid(row=1, column=1)
    password = Entry(RegisterFrame, font=('arial', 20), textvariable=PASSWORD, width=20, show="*")
    password.grid(row=2, column=1)
    firstname = Entry(RegisterFrame, font=('arial', 20), textvariable=FIRSTNAME, width=20)
    firstname.grid(row=3, column=1)
    lastname = Entry(RegisterFrame, font=('arial', 20), textvariable=LASTNAME, width=20)
    lastname.grid(row=4, column=1)
    btn_login = Button(RegisterFrame, text="Register", font=('arial', 18), width=25, command=Register,bg="cyan3",fg="black")
    btn_login.grid(row=6, columnspan=2, pady=20)
    lbl_login = Label(RegisterFrame, text="Login",  font=('arial', 17),bg="cyan3",fg="black")
    lbl_login.grid(row=0, sticky=W)
    lbl_login.bind('<Button-1>', ToggleToLogin)

def ToggleToLogin(event=None):
    RegisterFrame.destroy()
    LoginForm()

def ToggleToRegister(event=None):
    LoginFrame.destroy()
    RegisterForm()
```

```python
def Register():
    Database()
    if USERNAME.get == "" or PASSWORD.get() == "" or FIRSTNAME.get() == "" or LASTNAME.get == "":
        lbl_result2.config(text="Please complete the required field!", fg="orange")
    else:
        cursor.execute("SELECT * FROM `member` WHERE `username` = ?", (USERNAME.get(),))
        if cursor.fetchone() is not None:
            lbl_result2.config(text="Username is already taken", fg="red")
        else:
            cursor.execute("INSERT INTO `member` (username, password, firstname, lastname) VALUES(?, ?, ?, ?)", (str(USERNAME.get()), str(PASSWORD.get())
            conn.commit()
            USERNAME.set("")
            PASSWORD.set("")
            FIRSTNAME.set("")
            LASTNAME.set("")
            lbl_result2.config(text="Successfully Created!", fg="black")
        cursor.close()
        conn.close()

def Login():
    Database()
    if USERNAME.get == "" or PASSWORD.get() == "":
        lbl_result1.config(text="Please complete the required field!", fg="orange")
    else:
        cursor.execute("SELECT * FROM `member` WHERE `username` = ? and `password` = ?", (USERNAME.get(), PASSWORD.get()))
        if cursor.fetchone() is not None:
            lbl_result1.config(text="You Successfully Login", fg="blue")
            LoginFrame.destroy()
            park()
        else:
            lbl_result1.config(text="Invalid Username or password", fg="red")

LoginForm()
menubar = Menu(my_w)
```

```
def park():

    def load_and_preprocess_image():
        test_fldr='uploads'
        test_generator=ImageDataGenerator(rescale=1./255).flow_from_directory(
            test_fldr,
            target_size = (IMAGE_SIZE, IMAGE_SIZE),
            batch_size = 1,
            class_mode = None,
            shuffle = False)
        test_generator.reset()
        return test_generator

    def classify(model):
        batch_size = 1
        test_generator = load_and_preprocess_image()
        prob = model.predict_generator(test_generator, steps=len(test_generator)/batch_size)
        classified_prob = prob[0][0] if prob[0][0] >= 0.5 else 1 - prob[0][0]

        return classified_prob

    def logout():
         my_w.destroy()




    l1 = tk.Label(my_w,text='UPLOAD FILES & GET RESULTS',width=30,font=my_font1,bg='black',
                      fg='yellow',)
    l1.place(x=350, y=130, width=380)
    b1 = tk.Button(my_w, text='UPLOAD IMAGES',width=20,command = lambda:result(), activebackground='skyblue',font=('bold',17) ,bg='black',fg='yellow')
    b1.place(x=385,y=450, width=240, height=40)
    print(tf.__version__)

    titleLabel = Label(my_w, text='PARKINSONS DISEASE PREDICTION SYSTEM ', font=('italic', 25, 'bold '), bg='black',
                      fg='DarkTurquoise', )
    titleLabel.place(x=0, y=10,width=1100, height=80)


    b1 = tk.Button(my_w, text='Exit',command = lambda:logout(), activebackground='skyblue',font=('bold',17) ,bg='black',fg='yellow')
    b1.place(x=930,y=32)
```

```
model1 = load_model('model/Class1/model_Class1.h5',compile=False)
model2 = load_model('model/Class2/model_Class2.h5',compile=False)
model3 = load_model('model/Class3/model_Class3.h5',compile=False)


def result():
    filelist=glob.glob('uploads/all_class/*.*')
    for filePath in filelist:
        try:
            os.remove(filePath)
        except:
            print("Error While Deleting File")
    filename =upload_file()
    test_image2 = image.load_img(filename, target_size = (64, 64))
    test_image2 = image.img_to_array(test_image2)
    test_image2 = np.expand_dims(test_image2, axis = 0)
    result1 = model1.predict(test_image2)
    print(result1)
    result2 = model2.predict(test_image2)
    print(result2)
    result3 = model3.predict(test_image2)
    print(result3)
```

```
        if result1[0][0] == 1:
           if result2[0][0] == 1:
              if  result3[0][0] == 1:
                 prediction2='Parkinsons predicted'
                 value="Drug-Induced Parkinson"
              else:
                 prediction2='Parkinsons predicted'
                 value="Vascular Parkinson"
           else:
              prediction2='Parkinsons predicted'
              value="Idiopathic Parkinson"
        else:
           prediction2 = 'Normal'
           value="Normal"
        print(prediction2,value)
        prediction=prediction2
        values=value
        l2 = tk.Label(my_w,text="Result :  "+prediction,width=30,font=my_font1,bg='black',
                        fg='DarkTurquoise',)
        l2.place(x=115, y=500, width=400,height=55)
        l3 = tk.Label(my_w,text="Type  :  "+values,width=30,font=my_font1,bg='black',
                        fg='DarkTurquoise',)
        l3.place(x=475, y=560, width=440,height=55)
        shutil.copy(filename,UPLOAD_FOLDER)
        prob=classify(model1)

        l4 = tk.Label(my_w,text="Accuracy :  "+str(prob),width=30,font=my_font1,bg='black',
                        fg='DarkTurquoise',)
        l4.place(x=115, y=620, width=400,height=55)
        return filename


    def upload_file():
        filename=easygui.fileopenbox()
        print(filename)
        img=Image.open(filename)
        img=img.resize((200,140))
        img=ImageTk.PhotoImage(img)
        e1 =tk.Label(my_w)
        e1.place(x=380, y=180, width=240, height=250)
        e1.image = img
        e1['image']=img
        return filename


my_w.mainloop()
```

REFERENCES

[1] N. S. Hoang, Y. Cai, C.-W. Lee, Y. O. Yang, C.-K. Chui, and M. C. Heng Chua, "Gait classification for Parkinson's disease using stacked 2D and 1D convolutional neural network," in Proc. Int. Conf. Adv. Technol. Commun. (ATC), Oct. 2019, pp. 44–49, doi: 10.1109/ATC.2019.8924567.

[2] K. Hu, Z. Wang, S. Mei, K. A. E. Martens, T. Yao, S. J. G. Lewis, and D. D. Feng, "Vision-based freezing of gait detection with anatomic directed graph representation," IEEE J. Biomed. Health Informat., vol. 24, no. 4, pp. 1215–1225, Apr. 2020, doi: 10.1109/JBHI.2019. 2923209.

[3] G. Solana-Lavalle, J.-C. Galán-Hernández, and R. Rosas-Romero, "Automatic Parkinson disease detection at early stages as a pre-diagnosis tool by using classifiers and a small set of vocal features," Biocybern. Biomed. Eng., vol. 40, no. 1, pp. 505–516, Jan. 2020, doi: 10.1016/j. bbe.2020.01.003.

[4] S. Sivaranjini and C. M. Sujatha, "Deep learning based diagnosis of Parkinson's disease using convolutional neural network," Multimedia Tools Appl., vol. 79, nos. 21–22, pp. 15467–15479, Jun. 2020, doi:10.1007/s11042-019-7469-8.

[5] F. Aydın and Z. Aslan, "Recognizing Parkinson's disease gait patterns by vibes algorithm and Hilbert–Huang transform," Eng. Sci. Technol. Int. J., vol. 24, no. 1, pp. 112–125, Feb. 2021, doi: 10.1016/j.jestch. 2020.12.005.

[6] I. El Maachi, G.-A. Bilodeau, and W. Bouachir, "Deep 1D-convnet for accurate Parkinson disease detection and severity prediction from gait," Expert Syst.Appl.,vol. 143, Apr. 2020, Art. no. 113075, doi: 10.1016/j.eswa.2019.113075.

[7] B. Karan and S. Sekhar Sahu, "An improved framework for Parkinson's disease prediction using variational mode decomposition-Hilbert spectrum of speech signal," Biocybern. Biomed. Eng., vol. 41, no. 2, pp. 717–732, Apr. 2021, doi: 10.1016/j.bbe.2021.04.014.

[8] H. Gunduz, "An efficient dimensionality reduction method using filter based feature selection and variational autoencoders on Parkinson's disease classification," Biomed. Signal Process. Control, vol. 66, Apr. 2021,Art. no. 102452, doi: 10.1016/j.bspc.2021.102452.

[9] D. Yang, R. Huang, S.-H. Yoo, M.-J. Shin, J. A. Yoon, Y.-I. Shin, and K.-S. Hong,

“Detection of mild cognitive impairment using convolutional neural network: Temporal-feature maps of functional near-infrared spectroscopy,” FrontiersAgingNeurosci.,vol.12,p. 141, May 2020, doi: 10.3389/fnagi.2020.00141.

[10] S.-H. Yoo, S.-W. Woo, M.-J. Shin, J. A. Yoon, Y.-I. Shin, and K.-S. Hong, “Diagnosis of mild cognitive impairment using cognitive tasks: A functional near-infrared spectroscopy study,” Current Alzheimer Res., vol. 17, no. 13, pp. 1145–1160, Mar. 2021, doi: 10.2174/ 1567205018666210212154941.