

NEURODEGENERATIVE DISORDER DISEASE DETECTION USING VORTEX

Submitted in partial fulfillment of the requirements for the award of

Masters of Computer Science

By

H. NISHANTHI (42734013)



**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE AND HUMANITIES**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A++” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600119

APRIL 2024



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A++" Grade by NAAC | 12B Status by UGC I Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of H.NISHANTHI (42734013) who carried out the project entitled "NEURODEGENERATIVE DISORDER DISEASE DETECTION USING VORTEX" who carried out the project entitled under my supervision from JANUARY 2024 to APRIL 2024.

Internal Guide

MS. AISHWARYA.B, MCA

DEAN AND HEAD OF THE DEPARTMENT

Dr. REKHA CHAKRAVARTHI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 20.04.2024

Internal Examiner

External Examiner

DECLARATION

I am **H.NISHANTHI (42734013)** hereby declare that the Project entitled as **“NEURODEGENERATIVE DISORDER DISEASE DETECTION USING VORTEX”** done by me under the guidance of **MS.AISHWARYA.B, M.C.A** Assistant Profossor,Department of computer science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**,Jeppiaar Nagar,Rajiv Salai,Chennai-600119 is submitted in partial fulfillment of the requirements for the award of Masters of Science degree in Computer Science.

DATE: 20.04.2024

NISHANTHI.H

PLACE: CHENNAI

Signature of the Candidate

II

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.REKHA CHAKRAVARTHI M.E., Ph.D., Dean Of Arts and science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.AISHWARYA.B, MCA., Assistant Professor Department of computer science** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science who were helpful in many ways for the completion of project

NISHANTHI.H

ABSTRACT

Neurodegenerative disorder is a progressive disease, which is characterized by Various symptoms. Progressive neurodegenerative disorder affects nervous system in the elderly, which is characterized by motor symptoms such as tremor, rigidity, slowness of movement and problems with gait. In this work, an attempt has been made to classify the spiral images of healthy control and affected .The Vision based Convolutional Neural Network architecture is used to refine the diagnosis of neurodegenerative disorder disease. Vision Based methods have reported promising results in providing better characterization of PD in the early stages and are expected to have better sensitivity than standard clinical measures.

This project proposes a Vision Based novel deep learning architecture for neuro generative disorder screening.

Keywords: Neurodegenerative disease, deep learning, TensorFlow, CN

TABLE OF CONTENT

SL.NO	TITLE	PAGE.NO
01.	INTRODUCTION 1.1 DATA SCIENCE 1.2 ARTIFICIAL INTELLIGENCE 1.3 NATURAL LANGUAGE PROCESSING (NLP) 1.4 MACHINE LEARNING	1
02.	EXISTING SYSTEM 2.1 DRAWBACKS	12
03.	PROPOSED SYSTEM 3.1 ADVANTAGES	12
04.	LITERATURE SURVEY	14
05.	DATAFLOW DIAGRAM	16
06.	SYSTEM ARCHITECTURE	18
07.	WORKFLOW DIAGRAM	19
08.	USECASE DIAGRAM	20
09.	CLASS DIAGRAM	21
10.	ACTIVITY DIAGRAM	22
11.	SEQUENCE DIAGRAM	23
12.	E.R – DIAGRAM	24
13.	COLLABORATION DIAGRAM	25

14.	ENVIRONMENT REQUIREMENT 14.1 SOFTWARE REQUIREMENTS 14.2 HARDWARE REQUIREMENTS	26
15.	LIST OF MODULES	26
16.	MODULE DESCRIPTION 16.1 IMPORT THE GIVEN IMAGE FROM DATASET 16.2 TO TRAIN THE MODULE BY GIVEN IMAGE DATASET 16.3 WORKING PROCESS OF LAYERS IN CNN MODEL 16.4 CLASSIFICATION IDENTIFICATIO	28

LIST OF FIGURES

SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	18
02	WORKFLOW DIAGRAM	19
03	USECASE DIAGRAM	20
04	CLASS DIAGRAM	21
05	ACTIVITY DIAGRAM	22
06	SEQUENCE DIAGRAM	23
07	ER – DIAGRAM	24

1. INTRODUCTION

1.1 DATA SCIENCE:

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.

1.2 ARTIFICIAL INTELLIGENCE:

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of “intelligent agents” any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term “artificial intelligence” to describe machines that mimic “cognitive” functions that humans associate with the human mind, such as “learning” and “problem solving”, however this definition is rejected by major AI researchers.

AI programming focuses on three cognitive skills: learning, reasoning and selfcorrection.

Learning processes.

This aspect of AI programming focuses on acquiring data and creating rulesfor how to turn the data into actionable information.

The rules, which are called algorithms, provide computing devices with step- step instructions for how to complete a specific task.

This aspect of Artificial Intelligence programming is to focus on choosing the right algorithm to reach a desired outcome.

Self-correction processes.

This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible. AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repeti detail-oriented tasks like analyzing large numbers

of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

1.3 Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language_user_interfaces and the acquisition of knowledge directly from humanwritten source such as newswire texts. Some straightforward applications of natural language processing include information_retrieval, text mining, question answering and machine_translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. “Keyword spotting” strategies for search are popular and scalable but dumb”.

1.4 MACHINE LEARNING:

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three

categories. There are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning program is both given the input data and the corresponding labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

2. EXISTING SYSTEM:

The Brain MRI images are trained and tested to give the accuracy Disease measures on volumetric analysis is one of the widely used MRI protocols to demonstrate pathological modifications related to PD in the striatal region.

2.1 Drawback:

- They system does not give the accurate results .
- MRI Imaging involves high cost of production.
- The image resolution is low so the face expression will not be detected.

3. PROPOSED SYSTEM:

Neurodegenerative s disease (PD) is a progressive neurodegenerative disorder of the nervous system in the elderly. Vision Based methods have reported promising results in providing better characterization of PD in the early stages and are expected to have better sensitivity than standard clinical measures.

This project proposes a Vision Based novel deep learning architecture for neuro generative disorder screening.

In this project, analysis of MRI images for discrimination of healthy control and NDD (Neurodegenerative disorder) subjects is attempted using CNN model.

The proposed FAST-RCNN exploits Feature Extraction to tackle multi-view data from the Spiral Image data.

In training, the proposed model employs a data enhancement technology called SCI-KIT' Image Data Generator API on multi-view data.

The data features are enriched by this data augmented technology, which can increase the diversity of the experimental samples.

3.1 Advantages:

- They recognized faster and more accurately
- The model is trained to learn the low level to high level features and the classification results are validated.
- Thus with a rapid growth in the deep learning architectures, an objective diagnosis of Neurodegenerative 's disease will no longer be a laborious job for the clinicians in the near future.

4.LITERATURE SURVEY

General

Review of Literature Survey

Paper Title: Detecting Neurodegenerative 's Disease Using Hand Movements Captured by Leap Motion Sensor

Author: Anastasia Moskova et al.

Explanation: Employed machine learning algorithms (KNN, SVM, Random Forest, Decision Tree) achieving an overall accuracy of 95.3%.
Paper Title: Comparative Analysis of Machine Learning Algorithms for Neurodegenerative 's Disease Detection

Title: Neurodegenerative 's Disease Detection Using K Mean Clustering and Decision Tree Algorithms

Author: Shrihari K Kulkarni et al.

Explanation: Utilized various algorithms (XGBoost, CNN, ANN, Logistic Regression, Random Forest, SVM, Boosted Trees, RNN) with an overall accuracy of 94.5%.Paper Employed K Mean Clustering and Decision Tree algorithms with an overall accuracy of 85%.Paper Title: Analysis of Spiral and Wave Drawings for Neurodegenerative 's Disease Detection

Title: Detection of Neurodegenerative 's Disease from Hand-Drawn Images using Machine Learning Algorithms

Author: Sabyasachi Chakraborty et al.

Explanation: Utilized Logistic Regression, Random Forest Classifier, and CNN algorithms achieving an overall accuracy of 93.3%.Paper

Title: Tremor and Voice Analysis for Neurodegenerative 's Disease Detection

Authors: Ferdib-Al-Islam and Laboni Akter

Explanation: Employed various algorithms (Decision Tree, Gradient Boosting, KNN, Random Forest, HOG feature descriptor, Logistic Regression) with an overall accuracy of 89.33% in Tremor and Voice Analysis for Neurodegenerative 's Disease Detection Conducted tremor and voice analysis utilizing KNN, SVM, Naïve Bayes, and MRMR Feature Selection algorithms with an overall accuracy of 99.8%.Paper Title: XGBoost Algorithm for Voice Analysis in Neurodegenerative 's Disease Detection

Authors: Priyadharshini et al.

Explanation: Employed XGBoost algorithm for voice analysis achieving an accuracy of 92.76%.Paper Title: Machine Learning Techniques for Speech Datasets in Neurodegenerative 's Disease Detection

Title: Analysis of Voice Datasets for Neurodegenerative 's Disease Detection

Author: Basil K Varghese et al.

Explanation: Investigated speech datasets using SVR, Decision Tree Regression, Linear Regression, and SVM with an overall accuracy of 92.9%.Paper in Analysis of

Voice Datasets for Neurodegenerative 's Disease Detection Timothy
Wroge et al.

Analyzed voice datasets using machine learning algorithms Trees, Random achieving an overall accuracy of 86%.

5. OUTLINE OF THE PROJECT

- Define a problem
- Gathering image data set
- Evaluating algorithms
- Detecting results

DATAFLOW DIAGRAM:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to system designs. Symbols and Notations Used in DFDs Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

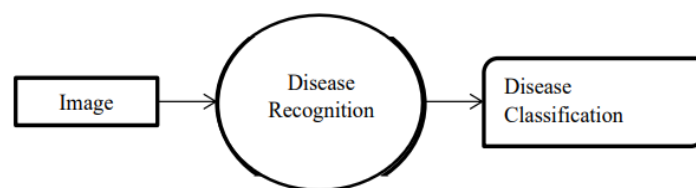
External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

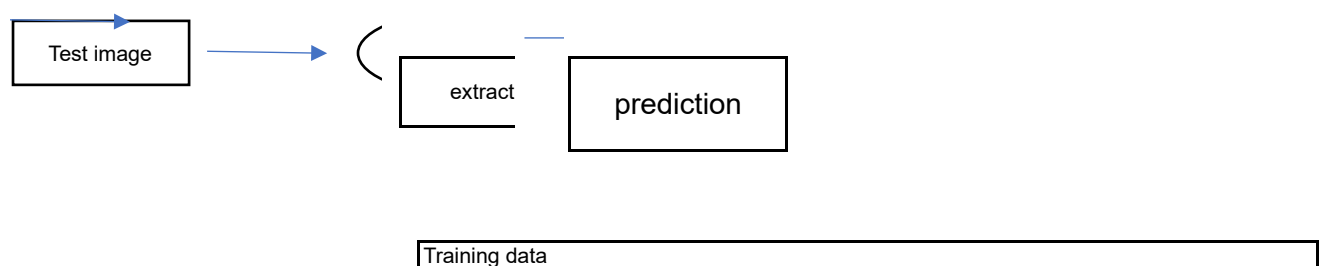
Data store: files or repositories that hold information for later use, such as a database table or a membership form.

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

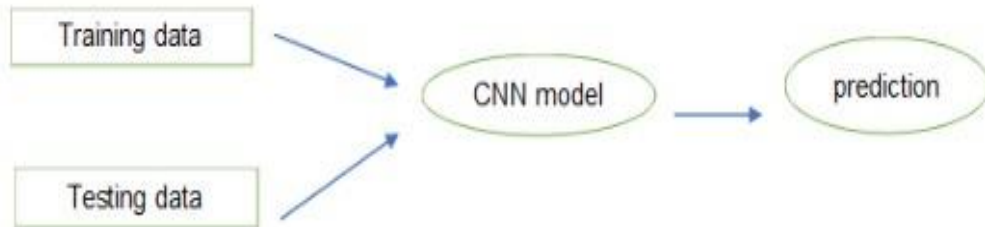
Dataflow levels:



Level 01



Level 02



Level 03

6. SYSTEM ARCHITECTURE

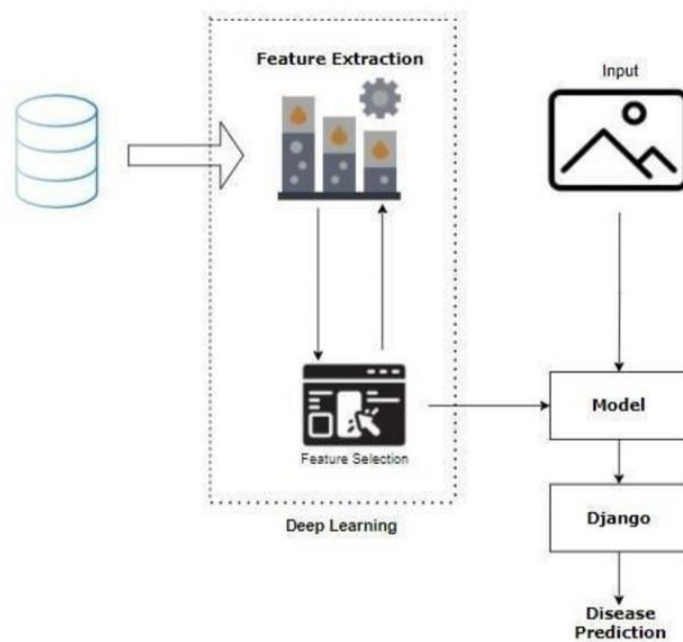


Fig.1 System Architecture

7. WORKFLOW DIAGRAM:

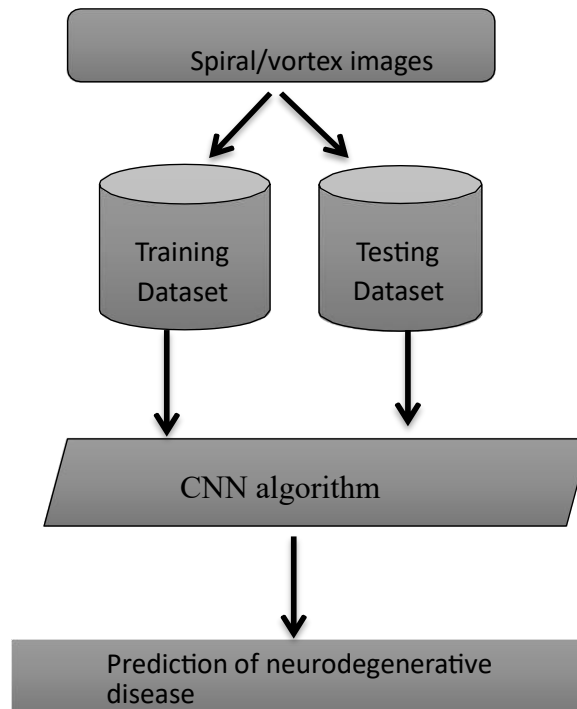


Fig.2 Workflow Diagram

8. USECASE DIAGRAM:

Use case diagrams are considered for high level requirement analysis of system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

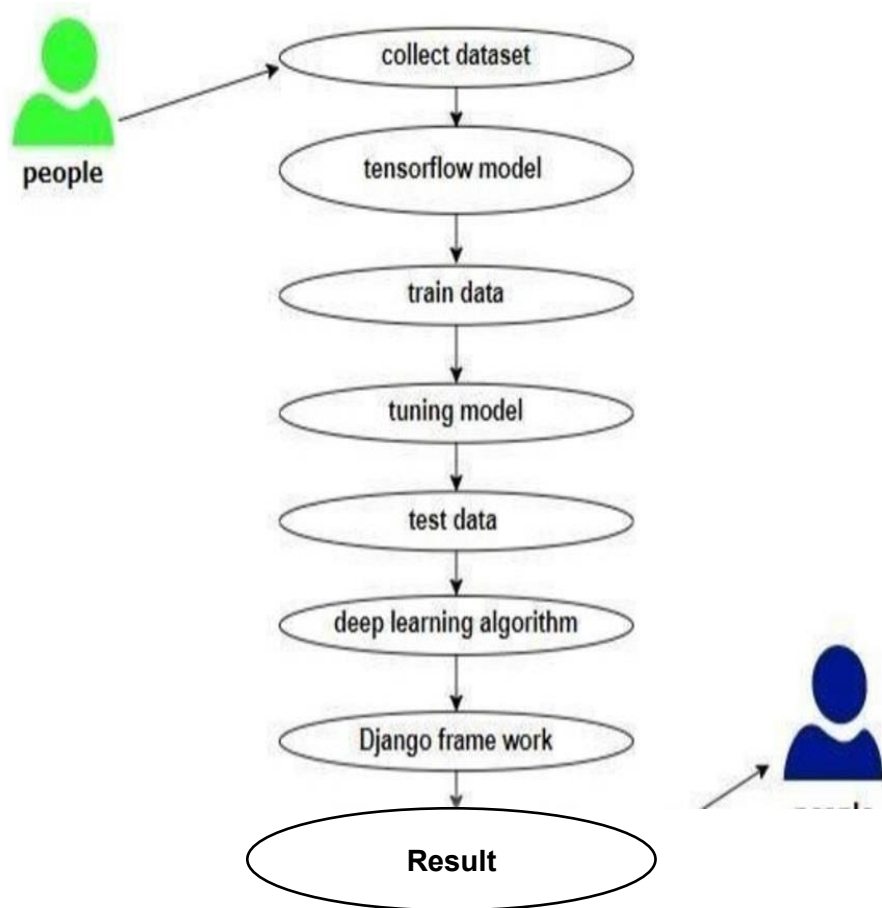


Fig.3 Use case Diagram

9. CLASS DIAGRAM:

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated.

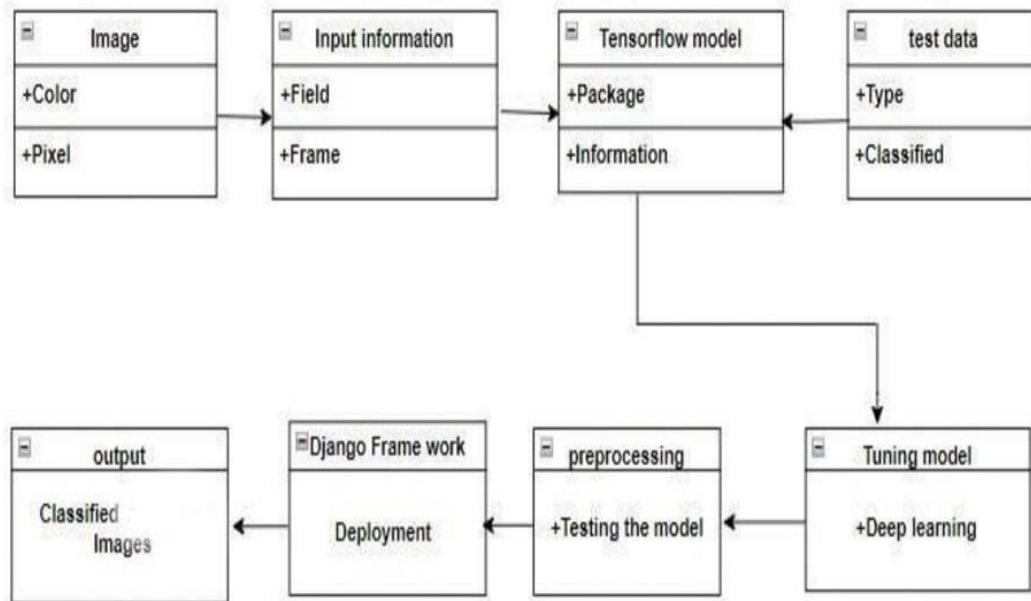


Fig.4 Class diagram

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

10. ACTIVITY DIAGRAM:

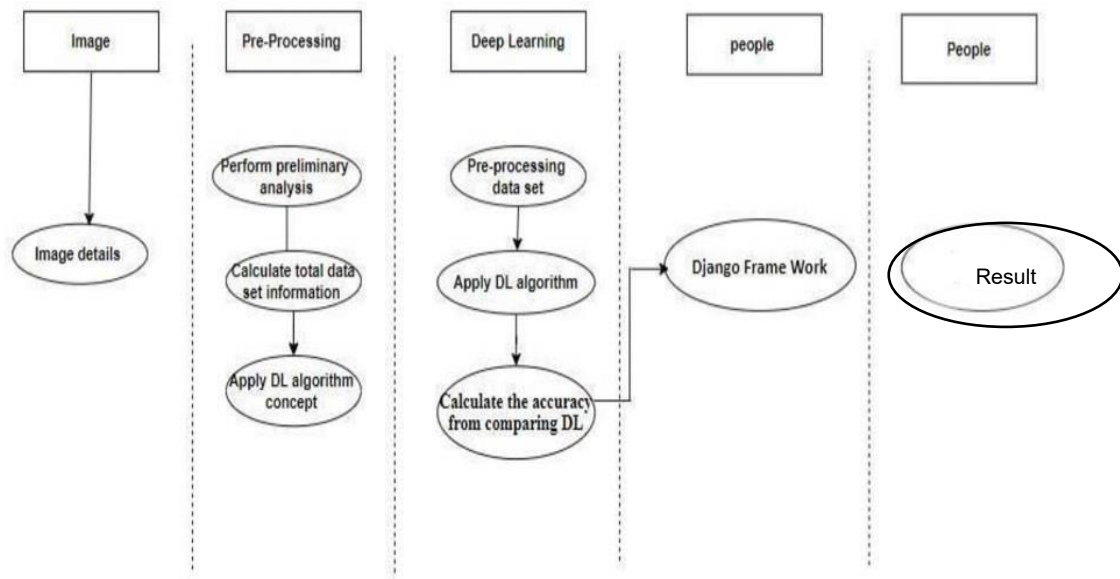


Fig.5 Activity diagram

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

11. SEQUENCE DIAGRAM:

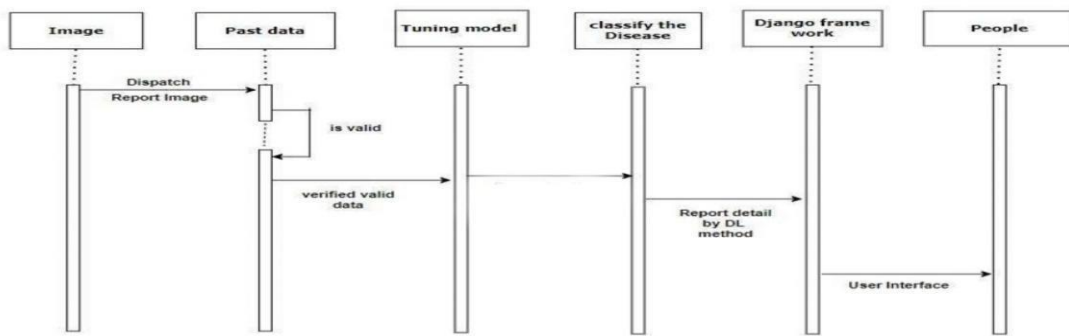


Fig.6 Sequence diagram

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important designlevel models for modern business application development.

12. ER DIAGRAM:

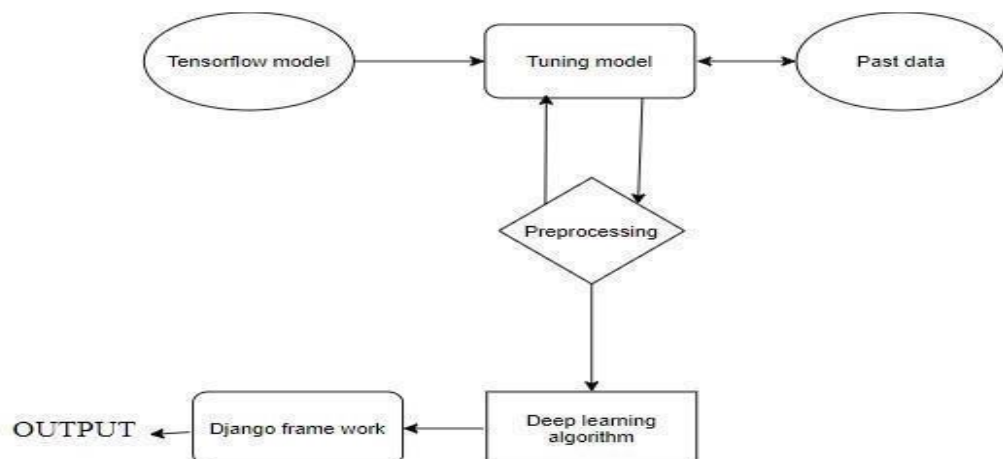
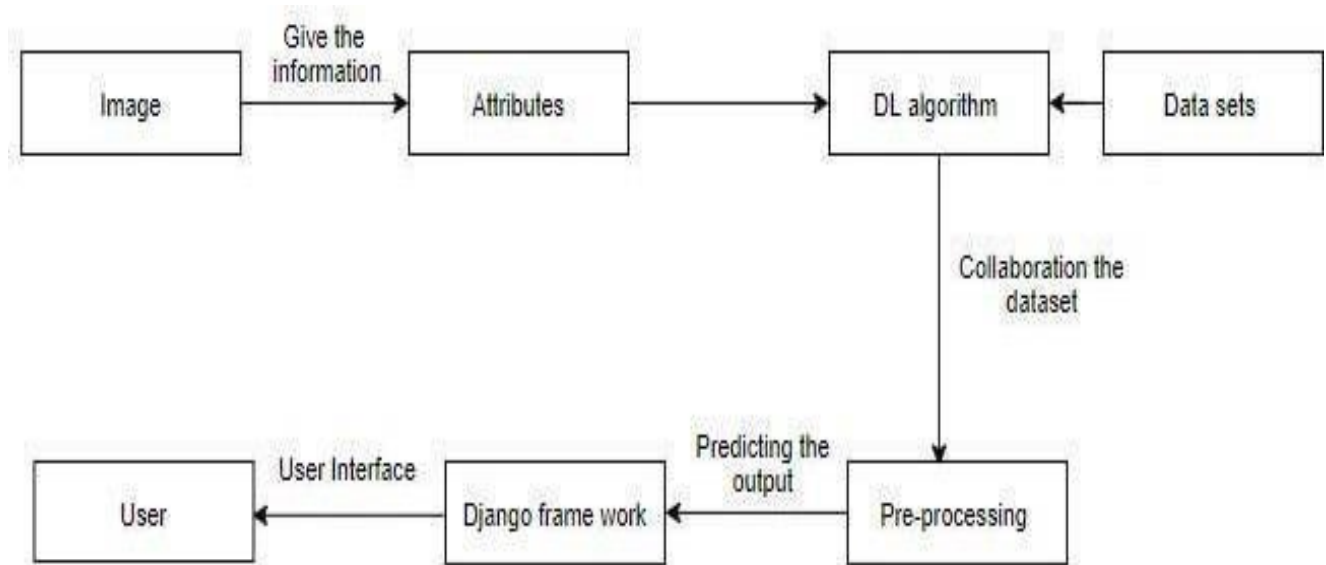


Fig.7 ER diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

13. COLLABORATION DIAGRAM:



A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction. The

collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram. it can be the decomposition of a use case, use case diagram, or part of a use case diagram. The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

14. ENVIRONMENTAL REQUIREMENTS:

14.1. Software Requirements

Operating System : WindowsLinux

Simulation Tool : Anaconda with
Jupyter Notebook

14.2. Hardware requirements

Processor : Pentium IV/III

Hard disk : minimum 8GB

RAM : minimum 2 GB

15. LIST OF MODULES:

1. Data Collection
2. Data Augmentation
3. Model Creation
4. Model Training and Testing

15.1 METHODOLOGY

Preprocessing and Training the model (CNN): The dataset is pre-processed such as Image reshaping, resizing and conversion to an form of array . Similar processing is also

done on the test image. A dataset consisting of about images of neurodegenerative disease images, out of which any image can be used as the test image for the software.

The train dataset is used to train the model (CNN) so that it can identify the test image and the disease it has CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D.

After the model is trained successfully, the software can identify the disease if the neurodegenerative image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the disease.

CNN MODEL STEPS:

CONV2D:

The 2D convolution is a fairly simple operation at heart: you start with a kernel, which is simply a small matrix of weights. This kernel “slides” over the 2D input data, performing an elementwise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are essentially, the weighted sums (with the weights being the values of the kernel itself) of the input features located roughly in the same location of the output pixel on the input layer.

Whether or not an input feature falls within this “roughly same location”, gets determined directly by whether it’s in the area of the kernel that produced the output or not.

This means the size of the kernel directly determines how many (or few) input features get combined in the production of a new output feature.

MaxPooling2D layer

Down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool size) for each channel of the input. The window is shifted by strides along each dimension.

The resulting output, when using the "valid" padding option, has a spatial shape (number of rows or columns) of: $\text{output_shape} = \text{math.floor}((\text{input_shape} - \text{pool_size}) / \text{strides}) + 1$ (when $\text{input_shape} \geq \text{pool_size}$)

The resulting output shape when using the "same" padding option is: $\text{output_shape} = \text{math.floor}((\text{input_shape} - 1) / \text{strides}) + 1$

FLATTEN LAYER

It is used to flatten the dimensions of the image obtained after convolving it. Dense: It is used to make this a fully connected model and is the hidden layer. Dropout: It is used to avoid over fitting on the dataset and dense is the output layer contains only one neuron which decide to which category image belongs.

Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, Dense layer

Dense implements the operation: $\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$ where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True). These are all attributes of Dense.

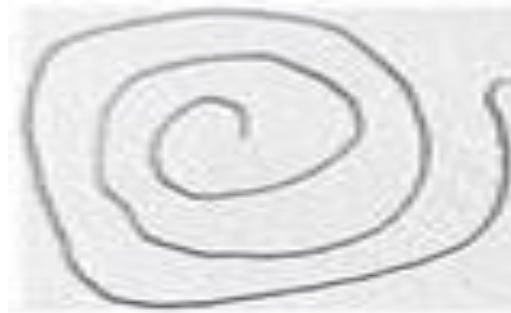
DROPOUT LAYER

The Brain MRI images are trained and tested to give the accuracy Disease measures on volumetric analysis is one of the widely used MRI protocols to demonstrate pathological modifications related to PD in the striatal region.

16.MODULE DESCRIPTION:

16.1 IMPORT THE GIVEN IMAGE FROM DATASET:

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size



healthy vortex image

class mode from this function we have to train using our own created network by adding layers of CNN. The variation in visual appearance will enable us to train a computer vision plus machine learning algorithm to automatically detect Neurodegenerative disease.

Below is the figure of collected data samples,



healthy vortex image



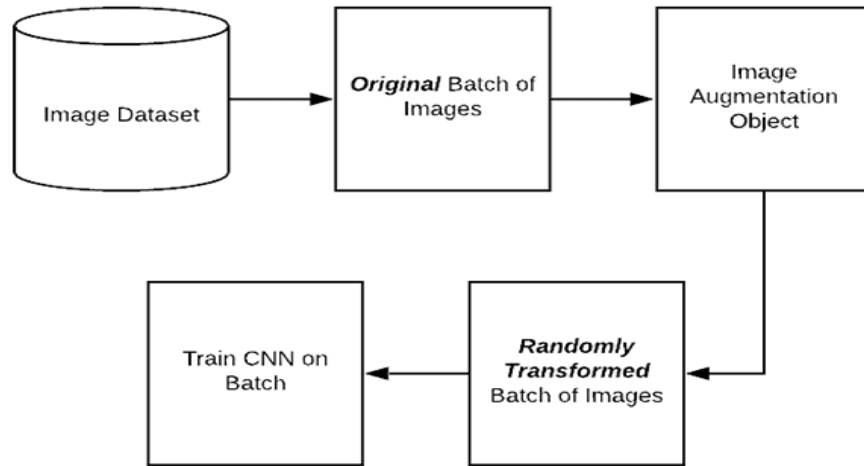
Disease affected vortex image

DATA AUGMENTATION:

Data augmentation is a technique commonly used in Deep Learning to artificially increase the amount of available data by generating additional data points. This approach involves applying geometric transformations, such as flipping and rotation, to the original images to create augmented versions.

The accuracy of deep learning models heavily relies on the quality, quantity, and contextual relevance of the training data. However, acquiring a sufficient amount of high-quality data can be a challenging task. It often requires substantial resources in terms of time and cost.

To address this limitation, manual image augmentation techniques were employed in our study. By applying flipping and rotating operations, we aimed to enhance both the quantity and quality of the data used to train our Convolutional Neural Network (CNN) model. This approach allowed us to artificially expand the dataset, providing the model with more diverse samples for improved learning and generalization.



MODEL CREATION:

We will utilize a Convolutional Neural Network (CNN) for training our model, which is a type of deep learning neural network capable of processing various types of data. CNNs are particularly effective in detecting patterns such as lines and gradients from input images.

In the case of Neurodegenerative 's disease, there exist several biomarkers that can be used to detect the disease. One such biomarker is the Spiral Drawing. Individuals with Neurodegenerative 's disease typically exhibit difficulty in drawing smooth and accurate spiral or wave diagrams. To train our model, we have gathered a dataset comprising both healthy and Neurodegenerative 's drawings. The dataset includes 98 Spiral Drawings for training, 28 for validation, and 14 for testing. Additionally, we have 91 Wave Drawings for training, 26 for validation, and 13 for testing. Prior to training, we pre-processed the images by resizing them to a resolution of 256x256.

For the prediction of results, the spiral models consist of 8 neural layers. These neural layers play a crucial role in analysing the input data and generating predictions based on the trained model

16.2 TO TRAIN THE MODULE BY GIVEN IMAGE DATASET:

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, We use validation data and validation steps to train our dataset.

Once we train the model with the training dataset, it's time to test the model with the test dataset. This dataset evaluates the performance of the model and ensures that the model can generalize well with the new or unseen dataset. The test dataset is another subset of original data, which is independent of the training dataset. However, it has some similar types of features and class probability distribution and uses it as a benchmark for model evaluation once the model training is completed.

Test data is a well-organized dataset that contains data for each type of scenario for a given problem that the model would be facing when used in the real world.

During the model training and testing phase, the input images were fed to the model in batches. In the case of the Spiral Model, each batch contained 14 images, while for the Wave Model, each batch contained 13 images. A total of 7 batches were used for training, 2 batches for validation, and 1 batch for testing.

For the training process, a sequential CNN model was employed with an input shape of (256,256,1), indicating the size and channels of the input images. The model utilized the sigmoid activation function and the Adam optimizer for optimization. The model was trained for 15 epochs, allowing it to learn from the data and update its parameters accordingly.

After training, the Spiral Model achieved an accuracy of 98% and a loss value of 0.0027. This indicates that the model performs highly accurately in classifying and predicting

whether an input image belongs to a healthy or Neurodegenerative 's affected individual based on their spiral drawings. On the other hand, the Wave Model achieved an accuracy of 84.61% with a loss value of 0.4070. This demonstrates the model's ability to distinguish between healthy and Neurodegenerative 's affected individuals based on wave drawings, although with a slightly lower accuracy compared to the Spiral Model.

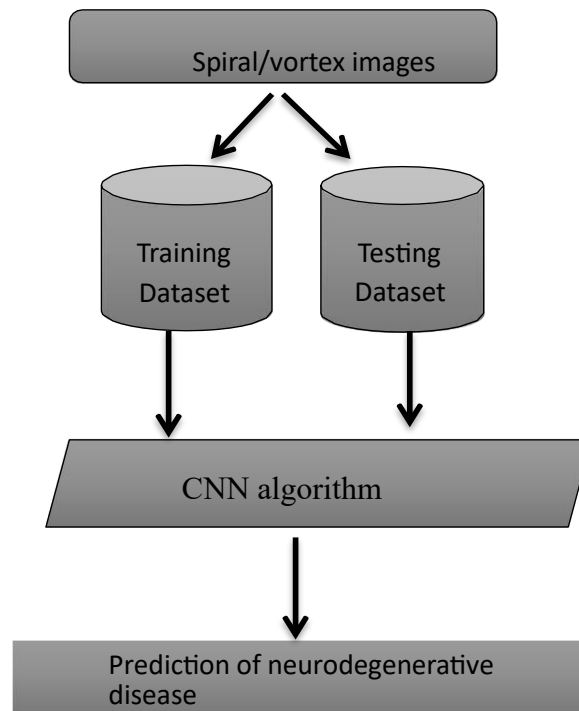
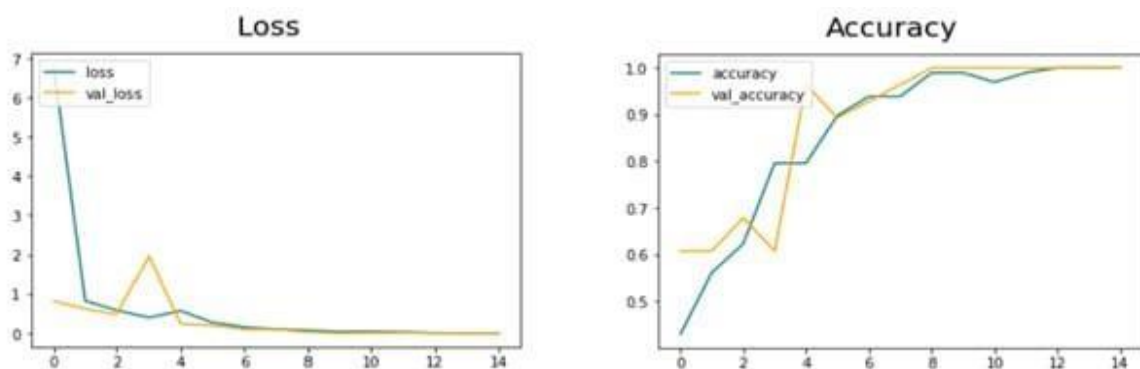


Fig.17.2.1 testing and training data



16.3 WORKING PROCESS OF LAYERS IN CNN MODEL:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

INPUT LAYER:

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension $28 \times 28 = 784$, it need to convert it into 784×1 before feeding into input.

CONVO LAYER:

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.

POOLING LAYER:

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducing to 2 x 2 dimensions.

FULLY CONNECTED LAYER (FC):

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

SOFTMAX / LOGISTIC LAYER:

SoftMax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and SoftMax is for multiclassification.

OUTPUT LAYER:

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

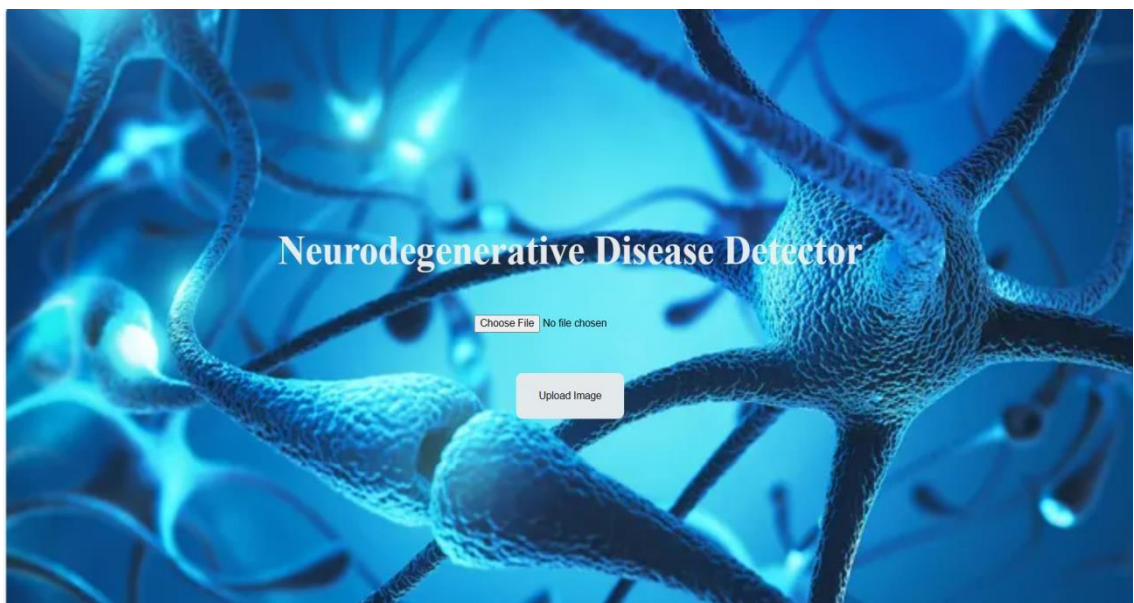
TO TRAIN THE MODULE BY GIVEN IMAGE DATASET

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.

Layer [type]	Output Shape	Parameters
conv1 [Conv2D]	(None, 128, 128, 128)	3328
max pooling2-4 (MaxPooling2)	(None, 40, 40, 128)	zero
conv2 [Conv2D]	(None, 40, 40, 64)	204864
max pooling2d-5 (MaxPooling2)	(None, 12, 12, 64)	zero
conv3 [Conv2D]	(None, 12, 12, 32)	18464
Max pooling2d-6 (MaxPooling2)	(None, 4, 4, 32)	zero
conv4 [Conv2D]	(None, 4, 4, 32)	9248
Max pooling2d-7 (MaxPooling2)	(None, 1, 1, 32)	zero
Flatten-1 [Flatten]	(None, 32)	zero
Dropout-2 [Dropout]	(None, 32)	zero
fc1 [Dense]	(None, 64)	2112
Dropout-3 [Dropout]	(None, 64)	zero
fc2 [Dense]	(None, 2)	130
238,146 parameters total		
238,146 trainable parameters		
Zero are non-trainable parameters.		

Fig:17.3 CNN Model Summary details

OUTPUT SCREEN:



CODING:

```
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Information Form</title>
<style> body { font-family:
Arial, sans-serif;
background-color: #f0f0f0;
margin: 0;
padding: 0;
}

h2 { text-align:
center; margin-
top: 50px; color:
#333;
}

form { max-width:
400px; margin: 0 auto;
background-color: #fff;
padding: 20px; border-
radius: 8px;
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

label {
font-weight: bold;
}

input[type="text"],
input[type="number"] {
width: 100%;
padding: 10px;
margin-bottom: 15px;
```

```
border: 1px solid #ccc;
border-radius: 4px;
  box-sizing: border-box;
}
```

```
input[type="submit"] {
width: 100%; padding: 10px;
background-color: #4CAF50;
  color: white;
border: none;
border-radius: 4px;
cursor: pointer;
font-size: 16px;
}
```

```
input[type="submit"]:hover {      background-
color: #45a049;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h2>Enter Your Information</h2>
```

```
<form action="front.html" method="post">
  <label for="name">Name:</label><br>
  <input type="text" id="name" name="name"><br>
  <label for="age">Age:</label><br>
  <input type="number" id="age" name="age"><br>
  <label for="phone">Phone Number:</label><br>
  <input type="text" id="phone" name="phone"><br><br>
  <input type="submit" value="Submit">
</form>
```

```
</body> </html> body {
margin: 0; padding: 0;
display: flex; justify-
content: center; align-
items: center; height:
100vh;
```

```
  background-image:
```

```
url('https://encrypted-
```

```

tbn0.gstatic.com/images?q=tbn:ANd9GcSsxLdAg2XREkFR4lQeP6-
OUpXbovm3UQBmg&s');
    background-size: cover;
}

```

```

h1 {
    color: #f1feff6; /* Adjust text color */    font-
size: 50px;
}

```

```

input[type="file"] {
display: block;
margin: 50px auto;
}

```

```

button {    font: size
50px;    display:
block;    margin: 0
auto;    padding: 20px
30px;
    background-color: #e4eaec; /* Adjust button color */
color: #0f0d0d;    border: none;    border-radius: 8px;
cursor: pointer;
}

```

```

#resultContainer {
margin-top: 80px;
font: size 90px;
color: #e4eaec
}

```

```

#resultContainer img {
max-width:    100%;
display: block;
    margin: 60px auto;

}

```

```

.hidden        {
display: none;
}

```

```

}
document.getElementById('uploadForm').addEventListener('submit',
function(event) {    event.preventDefault();

    var formData = new FormData();
    var        file        =        document.getElementById('imageInput').files[0];
    formData.append('image', file);

    // Here you can send the formData to your backend for processing

    // For now, let's just display the uploaded image
    var reader = new FileReader();    reader.onload =
function(event) {
    var img = document.getElementById('uploadedImage');
    img.src = event.target.result;
    document.getElementById('resultContainer').classList.remove('hidden');
}
    reader.readAsDataURL(file);
}); import os import cv2
import numpy as np from
skimage import feature
import randomimport
quantify_image(image):
    features = feature.hog(image, orientations=9,
                           pixels_per_cell=(10, 10), cells_per_block=(2, 2),
transform_sqrt=True, block_norm="L1")    return feadeftures def
load_split(path):
    # grab the list of images in the input directory, then initialize
    # the list of data (i.e., images) and class labels
    imagePath = list(path.list_images(path))
    data = []    labels = []
    # loop over the image paths
    for imagePath in imagePath:
        # extract the class label from the filename
        label = imagePath.split(os.path.sep)[-2]
        # load the input image, convert it to grayscale, and resize
        # it to 200x200 pixels, ignoring aspect ratio
        image = cv2.imread(imagePath)
        image    =    cv2.cvtColor(image,    cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image, (200, 200))
    # threshold the image such that the drawing appears as white

```

```

        # on a black background      image
= cv2.threshold(image, 0, 255,
                 cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
    # quantify the image
    features = quantify_image(image)
    # update the data and labels lists, respectively
data.append(features)      labels.append(label)
return (np.array(data), np.array(labels)) def
train_models(dataset):    # initialize the models
models = {
    "Rf": {
        "classifier": RandomForestClassifier(random_state=1),
        "accuracy": 0,
        "sensitivity": 0,
        "specificity": 0,
    },
    "Xgb": {
        "classifier": XGBClassifier(),
        "accuracy": 0,
        "sensitivity": 0,
        "specificity": 0,
    }
}
    # define the path to the testing and training directories
path = "../input/neurodegeneratives-drawings/" + dataset
trainingPath = os.path.sep.join([path, "training"])
testingPath = os.path.sep.join([path, "testing"])
    # load the data
    (trainX, trainY) = load_split(trainingPath)
    (testX, testY) = load_split(testingPath)
    # encode the labels    le =
LabelEncoder()    trainY =
le.fit_transform(trainY)
    testY = le.transform(testY)

    # train each model and calculate its metrics
    for model in models:
models[model]["classifier"].fit(trainX, trainY)
predictions = models[model]["classifier"].predict(testX)
cm = confusion_matrix(testY, predictions).ravel()    tn, fp,
fn, tp = cm

```

```

        models[model]["accuracy"] = (tp + tn) / float(cm.sum())
models[model]["sensitivity"] = tp / float(tp + fn)
models[model]["specificity"] = tn / float(tn + fp)    return
models def load_split(path):
    # Initialize the data and labels lists
    data = []
    labels = []

    # Define the ImageDataGenerator for data augmentation
    datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,      zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')

    # Loop over the image paths
    for label in os.listdir(path):
        label_path = os.path.join(path, label)
        for imagePath in os.listdir(label_path):
            # Load the input image, preprocess it, and augment
            image = cv2.imread(os.path.join(label_path, imagePath))
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            image = cv2.resize(image, (200, 200))
            image = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]
            image = np.expand_dims(image, axis=-1) # Expand dimensions for
compatibility with ImageDataGenerator # Generate augmented images
            augmented_images = datagen.flow(np.expand_dims(image, axis=0),
batch_size=1)      for batch in augmented_images:
data.append(quantify_image(batch[0])) # Quantify the augmented image
            labels.append(label)
            break # We only need one augmented image per original image

    return np.array(data), np.array(labels) from
keras.models import Sequential
from keras.layers import Dense, Flatten

def      create_model(input_shape):
model = Sequential()

```

```

    model.add(Flatten(input_shape=input_shape)) # Flatten layer to flatten the input
    shape
    model.add(Dense(128, activation='relu')) # Dense layer with 128 units and ReLU
    activation
    model.add(Dense(1, activation='sigmoid')) # Output layer with 1 unit and
    sigmoid activation for binary classification    return model

# Assuming input_shape is the shape of your input data
input_shape = (200, 200) # Adjust this according to your image dimensions model
= create_model(input_shape)
# Load and preprocess data
trainX,                                trainY                                =
load_split(os.path.join('C:\\Users\\hnish\\AppData\\Local\\Temp\\24137d5055b2-
4b4a-bf51-0f55e8203935_archive (2).zip.935\\archive.zip\\spiral\\training',
"training"))
testX,                                testY                                =
load_split(os.path.join('C:\\Users\\hnish\\AppData\\Local\\Temp\\24137d5055b2-4b4a-
bf51-0f55e8203935_archive (2).zip.935\\archive.zip\\spiral\\testing', "testing"))
# Load and preprocess data
trainX,                                trainY                                =
load_split(os.path.join('C:\\Users\\hnish\\AppData\\Local\\Temp\\24137d5055b2-
4b4a-bf51-0f55e8203935_archive (2).zip.935\\archive.zip\\spiral\\training',
"training"))
testX,                                testY                                =
load_split(os.path.join('C:\\Users\\hnish\\AppData\\Local\\Temp\\24137d5055b2-
4b4a-bf51-0f55e8203935_archive (2).zip.935\\archive.zip\\spiral\\testing',
"testing")) print("Random Forrest vs XGBoost
Classifier\\n\\n") for metric in ("accuracy",
"sensitivity", "specificity"):
print(f'{metric.capitalize()}: ")
    print("Random Forrest={:.2f}%, XGBoost={:.2f}% \\n".format(
spiralModels['Rf'][metric]*100, spiralModels['Xgb'][metric]*100)) def
test_prediction(model, testingPath):
    # get the list of images
    testingPaths = list(paths.list_images(testingPath))
output_images = []    # pick 15 images at random
for _ in range(15):
    image = cv2.imread(random.choice(testingPaths))
    output = image.copy()
    output = cv2.resize(output, (128, 128))
    # pre-process the image

```

```

image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
image = cv2.resize(image, (200, 200))      image = cv2.threshold(image,
0, 255,                                     cv2.THRESH_BINARY_INV |
cv2.THRESH_OTSU)[1]      # quantify the image and make predictions
based on the extracted features      features = quantify_image(image)
preds = model.predict([features])
label = "Neurodegeneratives" if preds[0] else "Healthy"

# draw the colored class label on the output image and add it to
# the set of output images
color = (0, 255, 0) if label == "Healthy" else (0, 0, 255)
cv2.putText(output, label, (3, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color, 2)
output_images.append(output)
plt.figure(figsize=(20, 20))      for i in
range(len(output_images)):
plt.subplot(5, 5, i+1)
plt.imshow(output_images[i])
plt.axis("off")
plt.show()

```

RESULT AND DISCUSSION:

The goal of the proposed approach is to detect the neurodegenerative disease via utilizing CCN based on hand-draw vortex images for differentiating the patients

(i.e., people who were diagnosed with the neurodegenerative disease) from the healthy

REFERENCES

- [1] Goldenberg, M.M. (2008). Medical management of Neurodegenerative's disease. *P&T*, 33(10): 590-606. [2] Neurodegenerative's Disease Symptoms. Mayo Clinic. <http://www.mayoclinic.org/diseasesconditions/parkinsonsdisease/basics/symptoms/con-20028488>, accessed on Sep. 13, 2017. [3] Jankovic, J. (2008). Neurodegenerative's disease: Clinical features and diagnosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 79(4): 368-376. <https://doi.org/10.1136/jnnp.2007.131045> [4] Prashanth, R., Roy, S.D., Mandal, P.K., Ghosh, S. (2014). Automatic classification and prediction models for early Neurodegenerative's disease diagnosis from SPECT imaging. *Expert Systems with Applications*, 41(7): 3333-3342. <https://doi.org/10.1016/j.eswa.2013.11.031> [5] Pham, H.N., Do, T.T.T., Chan, K.Y.J., Sen, G., Han, A.Y.K., Lim, P., Loon Cheng, T.S., Nguyen, Q.H., Nguyen, B.P., Chua, M.C.H. (2019). Multimodal detection of Neurodegenerative disease based on vocal and improved spiral test. In 2019 International Conference on System Science and Engineering (ICSSE), pp. 279-284. <https://doi.org/10.1109/ICSSE.2019.8823309> [6] Luo, C.Y., Chen, Q., Song, W., Chen, K., Guo, X.Y., Yang, J., Huang, X.Q., Gong, Q.Y., Shang, H.F. (2013). Resting-state fMRI study on drug-naive patients with Neurodegenerative's disease and with depression. *Journal of Neurology, Neurosurgery & Psychiatry*, 85(6): 675-683. <http://dx.doi.org/10.1136/jnnp2013-306237> [7] Neurodegenerative's Progression Markers Initiative. <http://www.ppmi-info.org/>, accessed on accessed on Sep. 13, 2017. [8] Rumman, M., Tasneem, A.N., Farzana, S., Pavel, M.I., Alam, M.A. (2018). Early detection of Neurodegenerative's disease using image processing and artificial neural network. In 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), IEEE, pp. 256-261. <https://doi.org/10.1109/ICIEV.2018.8641081> [9] Parisi, L., Ma, R.F., Zaernia, A., Yousseffi, M. (2021). Hyper-sinh-convolutional neural network for early detection of Neurodegenerative's disease from spiral drawings. *WSEAS Trans Comput. Res*, 9: 1-7. <http://dx.doi.org/10.37394/232018.2021.9.1> [10] Marras, C., Beck, J.C., Bower, J.H., Roberts, E., Ritz, B., Ross, G.W., Abbott, R.D., Savica, R., Van Den Eeden, S.K., Willis, A.W., Tanner, C.M. (2018). Prevalence of Neurodegenerative's disease across North America. *NPJ Neurodegenerative's Disease*, 4: 21. <https://doi.org/10.1038/s41531-018-0058-0> [11] Acharya, U.R.,

Oh, S.L., Hagiwar, Y., Tan, J.H., Adeli, H. (2018). Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signal

