ALZHEIMER'S DISEASE PREDICTION SYSTEM

PROJECT REPORT

Submitted by

NANDHANA P G	**(714020104053)**
PAUL DEEPAK S	**(714020104060)**
SIDDARTH S	**(714020104085)**
SNEKA T	**(714020104087)**

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SRI SHAKTHI

INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution,

Accredited by NAAC with "A" Grade

APRIL 2024

BONAFIDE CERTIFICATE

BONAFIDE CERTIFICATE

Certified that this project report **"ALZHEIMER'S DISEASE PREDICTION SYSTEM"** is the bonafide work of **"NANDHANA PG (714020104053), PAUL DEEPAK S (714020104060), SIDDARTH S (714020104085) AND SNEKA T (714020104087)"**, who carried out the work under my supervision.

SIGNATURE

Dr. K.E. KANNAMMAL

Professor and Head,

Department of CSE,

Sri Shakthi Institute of

Engineering and Technology,

Coimbatore- 641 062.

SIGNATURE

Dr. K.E. KANNAMMAL

Supervisor,

Professor & Head of CSE Dept,

Sri Shakthi Institute of

Engineering and Technology,

Coimbatore- 641 062.

Submitted for the University project work viva-voce Examination held on……

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

First and foremost, I would like to thank God Almighty for giving me the strength. Without his blessings this achievement would not have been possible.

We express our deepest gratitude to our **Chairman Dr. S. Thangavelu** for his continuous encouragement and support throughout our course of study.

We are thankful to our **Secretary Er. T. Dheepan** for his unwavering support during the entire course of this project work.

We are also thankful to our **Joint Secretary Mr. T. Sheelan** for his support during the entire course of this project work.

We are highly indebted to **Principal Dr. D. Elangovan** for his support during the tenure of the project.

We are deeply indebted to our **Head of the Department**, Computer Science and Engineering, **Dr. K.E. Kannammal**, for providing us with the necessary facilities.

It's a great pleasure to thank our **Project Guide Dr. K. E. Kannammal,** for her valuable technical suggestions and continuous guidance throughout this project work.

We are also thankful to our **Project Coordinators Mr.E.Subramanian** and **Ms.M.Mohanapriya** for providing us with necessary facilities and encouragement.

We solemnly extend our thanks to all the teachers and non-teaching staff of our department, family and friends for their valuable support.

NANDHANA P G
PAUL DEEPAK S
SIDDARTH S
SNEKA T

ABSTRACT

ABSTRACT

Deep learning, a state-of-the-art machine learning approach, has demonstrated exceptional performance compared to traditional machine learning in discerning intricate structures in complex high-dimensional data, particularly in computer vision. Recently, there has been significant interest in applying deep learning to the early detection and automated classification of Alzheimer's Disease (AD), driven by advancements in neuroimaging techniques that have produced large-scale multimodal neuroimaging data. Alzheimer's is a type of dementia, a brain disorder that typically affects individuals aged 60 and older but is increasingly affecting middle-aged people. Efforts are underway to control the disease using various techniques. One of the challenges in predicting AD using large datasets is feature extraction, as it can be challenging to identify the most relevant features for classification accurately. To address this challenge, a proposed approach involves using Convolutional Neural Networks (CNNs) for efficient classification and feature extraction. Feature extraction and selection are crucial factors for classification, and further research aims to improve performance by investigating optimal feature extraction and selection methods.

LIST OF FIGURES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LIST OF ABBREVIATIONS

AD	Alzheimer's Disease
MRI	Magnetic Resonance Imaging
fMRI	Functional Magnetic Resonance Imaging
CNN	Convolutional Neural Network
SPECT	Single-Photon Emission Computed Tomography
MMS	Mini Mental State Examination
FAQ	Functional Activity Questionnaire
SVM	Support Vector Machines
SGD	Stochastic Gradient Descent
ReLu	Rectified Linear Unit
RMSProp	Root Mean Square Propagation
ADAM	Adaptive Moment Estimation
PRS	Procedural Reasoning System
CDR	Clinical Dementia Rating
STBC	Space-Time Block Coding

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In the realm of healthcare, the accurate and timely diagnosis of brain diseases represents an ongoing challenge that requires a holistic understanding of the intricacies involved. Asthe prevalence of brain disorders continues to rise globally, the need for advanced diagnostic tools has become increasingly paramount. In this context, the integration of multiple modalities, such as neuroimaging data, clinical records, and cognitive assessments, has emerged as a promising approach to improve the accuracy and effectiveness of brain disease diagnosis. The synergy between these diverse data sources, when harnessed correctly, offers a multi-dimensional perspective that can lead to earlier detection, personalized treatment, and improved patient outcomes [1]. Brain diseases encompass a wide spectrum of conditions, including neurodegenerative disorders like AD, psychiatric illnesses like schizophrenia, and traumatic brain injuries.

AD stands as one of the most prevalent neurodegenerative disorders affecting millions worldwide. Initially described by Dr. Alois Alzheimer in 1906, this progressive and irreversible condition relentlessly erodes cognitive abilities, functional independence, and ultimately, one's sense of self. Despite extensive research and therapeutic advancements, the burden of AD continues to escalate, exerting profound emotional, societal, and economic tolls. Amidst this backdrop, the quest for early prediction emerges as a paramount imperative in combating this insidious ailment [2]. The journey of AD research traces back over a century, marked by pivotal discoveries elucidating its complex pathophysiology. Characterized by the aggregation of β-amyloid plaques and tau protein tangles in

the brain, AD precipitates synaptic dysfunction, neuroinflammation, and neuronal loss. These pathological hallmarks underpin the progressive cognitive decline and memory impairment emblematic of the disease. With an aging population and increasing life expectancy, the prevalence of AD has surged, amplifying its societal impact and underscoring the urgency for effective interventions.

While current diagnostic modalities primarily target symptomatic stages of AD, mounting evidence advocates for a paradigm shift towards early prediction. Recognizing the prodromal phase as a window of opportunity for intervention, researchers strive to identify biomarkers, genetic predispositions, and cognitive markers heralding impending cognitive decline. Early detection not only facilitates timely therapeutic interventions but also empowers individuals and caregivers with knowledge, fostering proactive lifestyle

modifications and advance care planning. Moreover, early prediction holds immense promise for accelerating the development of disease-modifying therapies, potentially altering the trajectory of AD progression. The present study endeavors to elucidate the intricacies of early AD prediction, encompassing diverse facets ranging from epidemiology and pathophysiology to diagnostic methodologies and intervention strategies. Through a comprehensive synthesis of existing literature and empirical insights, we aim to delineate the state-of-the-art in AD prediction, identify gaps in knowledge, and propose innovative approaches to advance the field [4]. Furthermore, by elucidating the ethical, societal, and public health implications of early prediction, this study seeks to engender informed discourse and catalyze concerted efforts towards combating the burgeoning epidemic of AD. The rationale for prioritizing early prediction of AD emanates from the recognition of the prodromal phase as a critical juncture for intervention. Mounting evidence suggests that pathological changes in the

brain precede the onset of clinical symptoms by several years, offering a window of opportunity for targeted interventions aimed at slowing or halting disease progression. By identifying individuals at heightened risk of developing AD, clinicians can implement personalized risk reduction strategies, monitor disease progression, and facilitate early enrollment in clinical trials for novel therapeutics. Moreover, early prediction holds profound implications for public health, enabling the implementation of preventive interventions and resource allocation strategies aimed at mitigating the burgeoning societal burden of AD.

In this modern era, since the overall life expectancy has increased, the rate of age related diseases occurrence has also increased. AD is an age related and common form of dementia which mostly affects elderly people [2]. AD is the permanent and progressive brain disease which slowly decrease memory, thinking, remembering, andreasoning skills [3]. There is a high possibility that Alzheimer disease increases progressively its' severity after the age of 65. Before AD becomes severe, it shows some warning signs like, poor judgment, changes in emotional behavior, difficulty to do familiar tasks, misplacing items, challengein solving problems and inability to learn new things [4]. Some of risk factors associated with AD are smoking, hypertension, age, diabetes, obesity and others. During recent decades, the number of patients has dramatically growing specially on developed countries which have high life expectancy. Currently, around 5.4 million Americans and 5million Europeans are affected by AD. The occurrence of AD is expected to quadruple by the year 2020[21] and estimated that, in 2050 one individual will develop the disease in every 30 seconds [22]. AD is being diagnosed using different techniques such as examination of the medical history, physical examination, laboratory test, neuropsychological or cognitive function testing and brain imaging scan. The diagnosis require specialist doctors such as

Psychiatrists, Neurologists and Psychologists [23]. Several kinds of diagnosis may involve and it will likely take more than a day to diagnose AD. The patients have to visit hospitals periodically to make periodic checkups.Currently, AD is also diagnosed using Machine Learning and agent system [1]. Machine Learning and agent system have made significant advances in the field of weatherforecasting, robotics, search engines, natural language processing, speech recognition, medical diagnosis, and handwriting recognition. Machine Learning, which is a core of Artificial Intelligence[24], is rapidly growing new technology[5] which used to design and develop classifiers that allow computers to “learn” [6]. This technology intends to solve problems of inference and prediction based on the available data and these data are helpful for decision making for human or intelligent computer system like an agent System. An agent system is "a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives”[17]. AnImportant aspect of agents is their ability to offer intelligence with interaction [18]. Intelligent agents have reactive, proactive and sociability properties. Intelligent agents have reactive, proactive and sociability properties. Multi agent system is a system which consists multiple agents which are working together either cooperatively or competitivelyto achieve their design objective. Various AD stage prediction has been proposed by many researchers using Machine Learning on different diagnosis techniques. These researches achieved their goal but someof them were working on AD using only diagnose with brain image scanning and the others require special devices that needs the healthcare center visit [11][13].

1.1 Problem Statement

The current challenge in AD research lies in the identification of reliable and accessible biomarkers for early detection. Traditional diagnostic methods often prove costly and invasive, limiting their widespread applicability. This study addresses this gap by proposing a novel approach based on Machine Learning techniques, specifically Support Vector Machines (SVM). While leveraging the accessibility and affordability of blood as a diagnostic medium, the research aims to identify a set of non-amyloid proteins that could serve as potential biomarkers for early-stage AD. The need for such biomarkers is paramount in advancing early intervention strategies and improving the efficacy of therapeutic interventions in the face of an aging population susceptible to neurodegenerative diseases. European e-health study shows that in each year, there are around 2.4 million unnecessary healthcare center visits by patients. Elderly people, including Alzheimer disease patients, are among those who are unnecessarily visiting the hospitals. AD patients can diagnose using different techniques but physical examination, laboratory test, and brain imaging scan require the physical appearance of the patient to the medical center. This results a large number of AD patients that visit the healthcare institutions. The increase in the number of visitors create a workload on professionals, high number of patients' queues and extra cost in terms of time and money on both the patients and health institutions. On each visit, the healthcare centers register and stored massive patients' data for future follow up. Usually these massive data are used only whenit is necessary to refer medical history of patients. Machine Learning creates another possibility to diagnosis AD patients using the massive stored cognitive function and medical history data.

1.2 Problem Objectives

The proposed system aims to leverage CNNs to addressthe challenges in early detection and automated classification of AD using large-scale multimodal neuroimaging data. The system focuses on improving the extraction and selection of relevant features fromcomplex datasets, overcoming existing limitations in traditional methods. The utilization of CNNs is expected to contribute significantly to the accuracy of AD classification, ultimately facilitating earlier detection and intervention. CNNs can automatically extract relevant features from complex neuroimaging data, eliminating the need for manual feature engineering, which is a time-consuming and error-prone process.This can lead to significant improvements in classification accuracy. The CNNs are very efficient at extracting features from images, making them well-suited for analysing the large amounts of data generated by neuroimaging studies. This efficiency can help to reduce the computational cost of AD classification and make it more feasible for real world application.

CHAPTER 2

LITERATURE REVIEW

In developed countries which have high life expectancy, the incidence of age related diseases rise exponentially with the increasing of age. AD is slowly progressive common form of dementia disorder which primarily affects the elderly population. This disease first affects the parts of the brain that control thought, memory and language. Elder people with AD may have trouble in remembering things which happened recently, names of people they knew, performing regular task and many more. Currently, about 4 million Americans and 5 million Europeans are affected byAD. The occurrence of AD is expected to quadruple by the year 2020[1] and it is estimated that, in 2050 one individual will develop the disease in every 30 seconds [2]. In America, AD is the one of the leading cause of death for elders around aged 65 years. Until these days, there is no cure for this disease. Many researchers are using various kinds of diagnosis and treatments to delay the stage progression.

Even though there's no specific test that confirms presence of AD, early and accurate diagnosis has a major impact on the development of AD stage change. In order to distinguish AD from other causes of memory loss, doctors typically rely on either one of or the combination of historical data, physical exam, cognitive testing, laboratory studies and brain imaging [19]. A person's medical history or historical data is one part of the assessment which is done by taking the person's AD related risk factors information including family history of AD, smoking, alcohol, diabetes, hypertension, heart disease, obesity (BMI) and gender [1][10][18]. Physical examination is a source of assurance that the patient

is as healthy as expected. Duringthis examination, the physician will check the blood pressure, temperature, pulse, lungs,heart beat and collect blood or urine samples for laboratory testing of the patient. The purpose of cognitive testing or neuropsychological testing is to investigate howthe respondent understands the questions and provide accurate answer. The well- known among these testing techniques are MMSE and FAQ. AD imaging is a brain diagnosis method to identify the person abnormalities. There are different kinds of brain imaging such as MRI, fMRI, PET and SPECT. These diagnosis techniques help to classify the person as a healthy or AD patient. The severity level of AD might differ between patients and it can be broadly divided into 5 stages: "No", "Questionable", "Mild", "moderate" and "severe".

On this information era, dealing with the available huge amount of raw informationis become one of the main problems. In order to process these mass of information and convert it into knowledge that people can use it, a potential data analysis technique, like Machine Learning, is necessary. Machine Learning (ML), which is a core of Artificial Intelligence [14], is rapidly growing new technology[5] which used to design and develop classifiers that allow computers to "learn" [16]. This technology enables a computer to analyze different size of data and learn which information is the most relevant from the specific dataset. Machine Learning has made significant advances in the field of weather forecasting, robotics, search engines, natural language processing, speech recognition, medical diagnosis, and handwriting recognition. ML aims to solve prediction and classification problems based on already existing data by learning the pattern. In ML there are four differentways of representing the structure: supervised learning, unsupervised learning, semi- supervised learning and reinforcement learning[6]. Among these, supervised and unsupervised learning

are prominently used [12]. The main difference between theseprominently used learning techniques is the availability of labeled examples or classified instances. Unlike in supervised learning, in unsupervised learning labeledexamples don't be included [6].

Supervised learning, also called classification **[11]**, aims to classify the new input efficiently and accurately. The classifier learns from a set of training data and corresponding classes in order to predict unlabeled instance[13][14]. Many researchers have been studying and using the most popular classification algorithmssuch as AdaBoost, C4.5, kNNs, decision trees, Naïve Bayes, neural networks and SVM [15][16][17]. Supervised learning is applicable in weather forecasting, predicting risk factors of disease in medical field, classificationof network packet in telecom and different other areas.

Unsupervised learning, also called clustering [8][4], takes a set of objects as an input and discovers a pattern if some of the sequences share the same object type. There are different unsupervised learning algorithms such as k-means, agglomerative clustering and Gaussian mixture model. The good clustering will collect similar objects together from the given environment near the leaf levels of the hierarchy and defer merging dissimilar subsets until near the root of the hierarchy[13]. Since the space of possible patterns that needs to be discovered is much larger, it makes unsupervised learning harder than supervised learning[9].

Agent system is a computer system which is capable of making autonomous action on its environment aiming to achieve its design objective. Agents perceive their environment using sensors to identify what is happening in the environment. Basedon the information acquired, the agents select and execute the appropriate action on the environment using their actuators. Reactivity,

proactiveness and sociability are important behaviors of intelligent agents. Beside these behaviors, intelligent agents might be built having a learning behavior as an additional feature. Agent technology is a combined results of computer science technologies such as artificial intelligence, object oriented programming and others. An Important aspect of agents is their ability to offer intelligence with interaction [8].

Multi-Agent System (MAS) is a system which consists multiple interacting agents that working together cooperatively or competitively to achieve their design objectives. Usually, multi agent system is used to solve problems which are difficultto be solved using single agent. In other words, MAS is an appropriate choice whenthe expertise and/or the knowledge required for solving the problems are distributed.The interaction between agents is not simply exchanging data; rather it includes cooperation, coordination and negotiation between agents as we do in our life.

Agent systems have been used and suggested by researchers to solve different problems raised across different spectrum of life. Some of them are: to handle emergency cases at hospitals [2], to assist patients and healthcare providers [1][2], to enhance conversation and text processing, for data mining in online social networks, for problem and solution modeling and for prediction , and for solving different problems in robotics, manufacturing ,business and economics. Agent architecture is a blueprint of an autonomous computer system (Agent). It specifies how the agents will be build. Its purpose is to show what component modules exist in the agent, how these modules works and how they interact to eachother. The components and their interactions designated by boxes and arrows whichindicate the data and control flows in the module. There are three types of agent architectures: symbolic/logical, reactive and hybrid architectures.

Deductive reasoning agent is a reasoning agent in which the agent behavior and its environment symbolically represented. The agent selects and executes the best actions by deducing these symbolic representations. Difficulty of translating the real world to symbolic description and complexity of symbolically representing real world entities and processes are the two key problems that should be solved in orderto build deductive reasoning agent. Another reasoning agent type is practical reasoning agent. It is based on the way how the human beings reason. Deliberation and means-ends reasoning are the fundamental activities in human practical reasoning. Deliberating is the process of deciding what state of affairs wants to achieve and means end reasoning is how to achieve that state of affairs. BDI and PRS architectures are found in this category of agent architecture. The problem of practical reasoning approach is deliberation and means end reasoning have cost in terms of time.

Reactive agents perceive their environment and automatically react for changes occurred on their environment without reasoning about them. Reactive agents madea decision based on the present changes regardless of what happened in past. Simplicity, robustness against failure and computational traceability can be mentioned among the advantages of reactive agents. The fundamental problems of reactive agents are: it is harder to engineer the agent when its number of behavior increases, designing of learning agent is difficult using this approach and others. These show that neither completely reactive nor completely deliberative way of agent building is not enough solution. Due to this, many researchers suggest using hybrid approach which combines both deliberative and reactive agent behaviors. Hybrid agents have hierarchy of agent component layers. Turing machines and InteRRaP architecture follow this approach. Turing machine is horizontal layered hybrid agent architecture. It has three layers:

reactive layer, planning layer and modeling layer. The reactive layer gives immediate response for the changes occurred in the environment. The planning layer has contains the agent proactive behavior. The modeling layerrepresents world entities. The embedded control subsystem decides which layer to have a control on the agent. InteRRaP is hybrid two-pass vertically layered agent architecture. It has three layers. The upper layer is responsible for social interaction of the agent, the middle layer is responsible for local planning and the lower layer is responsible for immediate reaction (reactivity). Each layer has associated knowledge base which contains appropriate world representation for the layer. Belief-Desire-Intention (BDI) architecture is prominent agent design architecture. It is based on the analogy of human mental stances: Belief, Desires and Intention. The agent's behavior abstracted in terms of these threemental stances [4]. In this architecture, the interpreter plays a fundamental role. The interpreter updates the belief of the agent constantly based on the information collected from the environment. Based on the agent belief, an appropriate desire and plan to be executed will be selected then the selected plan will be executed to achieve the intention of the agent. PRS is an agent architecture which consists explicitly represented mental stances those exist in BDI architecture. PRS architecture is the best established and durable implementation of BDIparadigm [6]. This architecture has been used for the implementation of major industrial multi-agent applications such as air-traffic control system at Sydney airport OASIS, SPOC business process management system and others. Subsumption architecture was invented by Rodney Brooks to address the shortcomings of the above mentioned ways of agent constructions. This architecture doesn't require a representation of world views in the memory [7]. In this architecture, the agent dynamically reacts to the events which occurred on its surrounding, accordingly.

This architecture is extremely effective in a situation where there is a limited memory capacity [8]. The agent behavior organized in the form of hierarchical layers. One advantage of this architecture is that the behavior of the agent can be easily changed byadding another behavior layer on the top layers[9].

Several researches have made study on different AD diagnosis data which is retrieved from MRI, fMRI, MMSE, PET, SPECT, demographic details and behavioral data. It is aimed to classify a person as AD patient or healthy using random forest algorithm. In order to meet their objective, the authors used a supervised learning that consists of five different steps. In the first step, the fMRI data are preprocessed to remove noise, missing values and errors from the dataset. In the next step, modeling of fMRI data is performed. Features, which are AD and healthy subjects, are extracted in the third step and then a subset of them is selected in the fourth step. Finally, a supervised random forests classification algorithm is applied on the data which consists of 41 subjects. The proposed method was evaluated using two different datasets. The first dataset consist healthy young, healthy old and demented subjects. The second dataset only consist healthy old anddemented subjects. Another research [2] has been studied by the same authors on similar fMRI data in five steps but the random forest classification algorithm has been modified using majority and weighted voting schemes. In recent work [2] of these researchers similar method, stage and dataset with their 41 subjects had used aiming to provide a supervised method to assist the diagnosis and monitoring the progression of AD using information which extracted from fMRI experiment.

Some researchers focus on retrieving different data from more than one diagnosis data for several reasons such as better classification and prediction accuracy. Sandhya Joshi et.al. [9] combined MMSE and FAQ tests in conjunction with four Machine Learning systems on a sample of 860 patients and control groups for their two goals i.e. to find an accuracy which save time and money in diagnosis of dementia and improving the sensitivity and specificity. Their model of classification of dementia states has four steps: data collection, data preprocessing, feature selection and classification. The objective of this model is to classify the various stages of dementia by applying a selected Machine Learning methods: C4.5, Naive Bayes and random forest classification. In addition to Machine Learning, neural network methods like multilayer perceptron was also applied for the evaluation of different stages of dementia. In their proposed system, the Machine Learning algorithms trained and tested on a sample of both MMSE and FAQ datasetseparately. The algorithms learn a description of four dementia MMSE classes (severe, mild cognitive, moderate AD and normal) as well as three dementia FAQ classes (severe, mild cognitive AD and normal) from the training sample of 860 subjects of each MMSE and FAQ. These different classes depicts by numbers knownas CDR score. This score is used to determine the presence or absence of dementia and also to find out patient's dementia severity stage. The CDR score of 0 indicates "No Cognitive Impairment" and CDR score of,0.5, 1, 2 and 3 shows "Questionable", "Mild", "Moderate" and "Severe dementia" respectively.

The author of [4] proposes an automated segmentation methods on MR brain volumes and classifier models for patient’s CDR score prediction. Their aim was predicting the CDR using Bayesian classifier on 371 subjects of MRI data. The classification was computed by the probability of each class given the particular

instance composed by the attributes and then selecting the class label with the highestprobability. The authors finally conclude that their idea is promising for diverse clinical information on computer-aided diagnosis application to accurately identify AD. A little more MRI data, that consists a cross-sectional collection of 218 subjects aged18 to 59 years and 198 subjects aged 60 to 96 years, was used in [4]. The authors proposed an automated novel method to classify of a person as Alzheimer patient orhealthy using MRI-scan data. This method combines independent component analysis and voxel of interest for classification with five steps. These steps are preprocessing of MRI data, segmentation of gray matter of the brain, decomposition using independent component analysis, extraction of voxel of interest, and classification by a SVM. After their experiment, the results showsthat the proposed method is able to provide better classification results than other related works that presented in their paper.

Having difficulties in organizing and planning things, exhibiting memory losses, forgetting names and words, making poor judgment, easily losing direction and having difficulties on speech are problems most of the time faced by AD patients ontheir early and moderate stages. Many researchers and computer scientists have proposed different computing methods and techniques to assist AD patients to enjoytheir normal life or at least partial of it by persisting these problems. Reminding patients for their medication or other activities and giving patients a hintto trigger their memory to remember forgotten things are form of assistances commonly found on systems proposed by computer scientists to assist AD patients. In [7] the authors suggested using of Ambient Intelligence (AmI) to support AD patients. In this paper the authors proposed a mobile application which gives a memory trigger for AD patients to remind them

sequenceof steps to prepare simple day to day activities. Furthermore in this system, significant deviation from their normal daily activities will be notified for the caregivers. The proposed system consists of three components: patient terminal, caregivers' station and server station. The patient terminal, which is a multi agent system, is responsible to collect and send patient's context information to the web server and to assist the patient based on the acquired information. On the web server, the data will be recorded, analyzed and processed. The caregiver gets information about the patients directly from the patient terminal or the server accordingly. Generally in the proposed system, the patient gets memory trigger assistance and getreminder for the task to be done. On the other hand, the caregivers benefited from the system by getting remote communication service with patients, up-to-date information about the patients whenever they required and alerting message when the patient is in dangerous situation.

In [11], the authors presented a user centered designing approach to develop cognitive prosthetic device which enhance AD patients' safety feeling and to alleviate social problems they are facing. From the workshop and conducted interview which carried out in Amsterdam, Lulea and Belfast on patients and caregivers by psychologists, the authors identified that maintaining social contacts,remembering, performing daily life activities and enhancing patients' safety feelingsare main areas of cognitive support needed by the patients. The authors implementeda prototype of a system which incorporates these features.

Patients' memory impairment increases progressively through time. Currently,it is impossible to cure memory impairment caused by dementia rather than delaying its progression. Delaying the impairment could be one sort of assistance which could be achieved using computation techniques and technologies. In [12], the

authors proposed a novel system which incorporate portable mini stationary bike toenable the patient to make physical exercise on it. The system also provides multiplechoice question game in visual and interactive fashion. The game with combinationof the physical exercise aimed to enhance some areas of patient's brain capacity suchas patents' memorizing capacity, judgment skill, recollection, matching capability and problem solving ability. While the patient driving the mini stationary bike, the system gives them multi choice question to answer it. In order to be successful in thegame, the patient has to answer 30 questions within the time the animation on the screen moves from starting point to the end point. By doing so, the patient simultaneously carried out physical and cognitive exercises enjoyably, which possibly delays patient's mental degradation.

VALIDATION OF A REGRESSION TECHNIQUE FOR SEGMENTATION OF WHITE MATTER HYPER INTENSITIES IN AD:

MahsaDadar et al (2019): Segmentation and volumetric quantification of white matter hyper intensities is essential in assessment and monitoring of the vascular burden in aging and AD, especially when considering their effect on cognition. Manually segmenting WMHs in large cohorts is technically unfeasible due to time and accuracy concerns. Automated tools that can detect WMHs robustly and with high accuracy are needed. A fully automatic technique forsegmentation and volumetric quantification of WMHs in aging and AD. The proposed technique combines intensity and location features from multiple MRI contrasts and manually labeled training data with a linear classifier to perform fast and robust segmentations.

STUDYING THE MANIFOLD STRUCTURE OF AD: A DEEP LEARNING APPROACH USING CONVOLUTIONAL AUTO ENCODERS:

Francisco J. Martinez-Murcia et al (2019): Many classical Machine Learning techniques have been used to explore AD, evolving from image decomposition techniques such as principal component analysis towardhigher complexity, non-linear decomposition algorithms. With the arrival of the deep learning paradigm, it has become possible to extract high-level abstract featuresdirectly from MRI images that internally describe the distribution of data in low- dimensional manifolds. The distribution of the extracted features in different combinations is then analyzed and visualized using regression and classification analysis, and the influence of each coordinate of the auto encoder manifold over thebrain is estimated.

MODELING DISEASE PROGRESSION VIA MULTISOURCE MULTITASK LEARNERS: A CASE STUDY WITH AD:

LiqiangNie et al (2016): Understanding the progression of chronic diseases can empower the sufferers in taking proactive care. To predict the disease status in the future time points, various Machine Learning approaches have been proposed. However, a few of them jointly consider the dual heterogeneities of chronic diseaseprogression. In particular, the predicting task at each time point has features from multiple sources, and multiple tasks are related to each other in chronological order.

LATENT REPRESENTATION LEARNING FOR AD DIAGNOSIS WITH INCOMPLETE MULTI-MODALITY NEUROIMAGING AND GENETIC DATA:

Tao Zhou et al (2019): The fusion of complementary information contained in multi-modality data [e.g., MRI, PET, and genetic data] has advanced the progress of automated AD diagnosis. However, multi-modality based AD diagnostic models are often hindered by the missing data, i.e., not all the subjects have completemulti-modality data.

One simple solution used by many previous studies is to discardsamples with missing modalities. However, this significantly reduces the number oftraining samples, thus leading to a sub-optimal classification model. The aim of this research is to investigate a Machine Learning algorithms for Alzheimer disease stage prediction and integrate the algorithm with multi agent system for accuracy improvement. The diagnosis of AD patients carries out by theirmedical history and cognitive function data using Machine Learning.

CHAPTER 3

METHODOLOGY

3.1 Existing System:

The existing system addresses the limitations of amyloid-based biomarkers in AD diagnosis by focusing on the identification of blood-based non-amyloid biomarkers for early detection. Leveraging Machine Learning techniques, particularly SVM, the system aims to create multivariable models capable of learning complex data patterns. Through novel feature selection and evaluation methods, the system has identified five panels of non-amyloid proteins with potential as biomarkers for early AD. Notably, the combination of A2M, ApoE, BNP, Eot3, RAGE, and SGOT emerges as a promising biomarker profile for early disease detection. This approach holds promise for improving AD diagnosis by leveraging accessible and cost-effective blood-based biomarkers to identify individuals at early stages of the disease process[40].

3.2 Proposed System:

The proposed system represents a significant advancement in the field of AD detection and classification, leveraging CNNs to analyze large-scale multimodal neuroimaging data. AD, a progressive neurodegenerative disorder, poses significant challenges in early detection and accurate classification, hindering timely interventions and treatment efficacy. Current diagnostic methods, primarily reliant on amyloid-based biomarkers and tests, have notable limitations, including their inability to provide comprehensive information about the disease process and their failure to identify individuals before significant amyloid-beta accumulation in the brain occurs.

Recognizing these limitations, the proposed system seeks to address these challenges by harnessing the power of CNNs, a type of deep learning model renowned for its effectiveness in image analysis tasks. CNNs are particularly well-suited for processing neuroimaging data due to their ability to automatically learn hierarchical representations of features, making them ideal for capturing subtle patterns indicative of AD pathology. By implementing CNNs, the system aims to enhance both the efficiency of classification and the extraction of relevant features, crucial components in accurately predicting and identifying AD.

One of the key advantages of CNNs lies in their ability to autonomously extract and select relevant features from complex neuroimaging datasets, eliminating the need for manual feature engineering and overcoming existing limitations in traditional methods. Through automated feature extraction, CNNs can effectively capture both local and global features within neuroimaging data, facilitating more accurate classification of AD. This automated approach not only improves classification accuracy but also streamlines the analysis process, reducing the burden of manual intervention and enabling healthcare professionals to focus on interpretation and clinical decision-making. The proposed system's utilization of CNNs holds significant promise for advancing early detection and intervention in AD. By accurately identifying individuals at risk of developing AD at earlier stages, the system facilitates timely interventions and treatments, potentially improving outcomes for affected individuals. Furthermore, the system's automated analysis of neuroimaging data contributes to a deeper understanding of AD pathology, uncovering novel insights into the disease's progression and underlying mechanisms. Despite its potential benefits, the proposed system also faces challenges and limitations. Further research and validation are necessary to fully assess the system's performance and reliability in real-world clinical settings. Additionally, ongoing advancements in deep learning and

neuroimaging technologies may offer opportunities for refining and enhancing the proposed system's capabilities in the future.

In summary, the proposed system represents a promising approach to addressing the challenges of AD detection and classification, leveraging CNNs to analyze large-scale multimodal neuroimaging data. By enhancing both the efficiency of classification and the extraction of relevant features, the system aims to facilitate earlier detection and intervention in AD, ultimately improving outcomes for affected individuals and advancing the understanding of the disease.

3.3 Proposed Plan of Work:

In this project, a dataset consisting of 6400 MRI scans collected from various sources has been utilized, categorized into four distinct classes: Mild Demented, Moderate Demented, Non Demented, and Very Mild Demented. Prior to model development, an analysis of the target value distribution was conducted to ensure a balanced representation of classes across the dataset. Following this assessment, the dataset was partitioned into training, testing, and validation subsets, facilitating the evaluation of the model's performance on unseen MRI scans.

The implementation of CNNs was chosen as the cornerstone of the modeling approach due to their exceptional capabilities in handling image data and extracting intricate features. CNNs excel in capturing both local and global patterns within images, making them ideally suited for tasks such as image classification. The training process involves feeding the CNN model with the training dataset, where it learns to discern relevant patterns and features associated with each class of MRI scans. Through successive iterations, the model adjusts its internal parameters to minimize the discrepancy between predicted and actual class labels. Once training is completed, the performance

of the model is evaluated using the testing dataset, allowing for an assessment of its ability to accurately classify unseen MRI scans. The validation dataset serves as an additional measure to assess the generalization capability of the trained model. By evaluating the model's performance on a separate set of unseen data, insights into its robustness and potential for real-world application are obtained. Ultimately, the success of the CNN model hinges on its ability to accurately predict the classes of MRI scans across all categories. Through rigorous evaluation and validation, the model's effectiveness in accurately classifying previously unseen MRI scans will be determined, providing valuable insights into its potential utility in clinical settings for aiding in the diagnosis and management of dementia-related conditions.

3.4 CONVOLUTIONAL NEURAL NETWORKS

AD is a progressive neurodegenerative disorder that primarily affects older adults, leading to memory loss and cognitive decline. Early detection of AD is crucial for timely intervention and management of the disease. CNNs have emerged as a powerful tool for analyzing medical imaging data, including brain images, to aid in the early diagnosis and prediction of AD. AD is characterized by the accumulation of amyloid plaques and tau tangles in the brain, which disrupt neuronal function and lead to brain atrophy. Neuroimaging techniques such as MRI and Positron Emission Tomography (PET) can visualize these changes in the brain, providing valuable information for the diagnosis and monitoring of AD progression. CNNs have shown great promise in analysing brain imaging data for the prediction of AD [38]. These deep learning models are capable of automatically learning hierarchical features from raw imaging data, which can then be used to classify images into different categories, such as AD, Mild Cognitive Impairment (MCI), or

healthy controls.

The architecture of CNNs used for AD prediction typically consists of multiple convolutional and pooling layers followed by fully connected layers. The convolutional layers extract spatial features from the input images, while the pooling layers reduce the spatial dimensions of the features, making the model more computationally efficient. The fully connected layers at the end of the network use these features to make predictions about the presence or progression of AD. CNNs for AD prediction are trained on large datasets of brain images, including images from AD patients, MCI patients, and healthy controls[39].

The model is trained to minimize a loss function, such as cross-entropy, which measures the difference between the predicted and actual labels. The model is then validated on a separate dataset to assess its performance and generalization ability. CNNs offer several advantages for AD prediction. They can automatically learn relevant features from brain imaging data without the need for manual feature extraction, which can be time-consuming and error-prone. CNNs can also handle large and complex datasets, making them well-suited for analyzing the vast amounts of imaging data generated in AD research [32].

1. **Input Layer**:

- The input layer defines the shape of the input data that the network will receive. In this case, the input shape is (128, 128, 3), indicating that the network expects images that are 128 pixels in height and 128 pixels in width, with 3 color channels (RGB).

2. The choice of input shape is crucial as it determines the size of the input tensor that will be passed through the network. Ensuring that the input shape matches the dimensions of the input data is essential for the network to process the images correctly.

3. **Convolutional Layers**:

- Convolutional layers are the building blocks of CNNs and are responsible for learning features from the input images.
- Each convolutional layer applies a set of filters to the input image. These filters slide across the input image, performing element-wise multiplication and summing the results to produce a feature map.
- The number of filters in each convolutional layer (16, 32, and 128 in this case) determines the depth of the feature maps and the number of unique features that can be learned.
- The choice of a 3x3 filter size is common in CNNs as it allows the network to capture small, local features in the input images.
- The 'relu' activation function is applied after each convolution operation. ReLU introduces non-linearity into the network, allowing it to learn complex patterns in the data.

4. **MaxPooling Layers**:

- MaxPooling layers are used to reduce the spatial dimensions of the feature maps while retaining the most important information.
- MaxPooling operates on small subregions of the input (e.g., 2x2 pixels) and outputs the maximum value from each subregion. This downsampling helps in reducing the computational complexity of the network and makes it more efficient.

5. **Flatten Layer**:

- The flatten layer is used to convert the 3D output of the convolutional layers into a 1D vector that can be fed into the dense layers.
- This flattening step is essential for connecting the convolutional part of the network to the fully connected part, as dense layers require 1D input vectors.

6. **Dense Layers**:

- Dense layers are fully connected layers where each neuron in a layer is connected to every neuron in the previous layer.
- The first dense layer has 128 units and uses the 'relu' activation function. This layer helps in learning high-level features from the flattened input.
- The second dense layer has 64 units and also uses the 'relu' activation function. It further refines the features learned by the previous layer.
- The final dense layer has 4 units, corresponding to the number of classes in the classification task. The 'softmax' activation function is used in this layer to convert the raw output into class probabilities, facilitating multi-class classification.

7. **Model Compilation**:

- The model is compiled using the Adam optimizer, which is an adaptive learning rate optimization algorithm that is well-suited for training deep neural networks.
- The choice of optimizer is crucial as it determines how the model updates its weights based on the loss function during training.
- The loss function used is 'sparse_categorical_crossentropy', which is suitable for multi-class classification tasks where the target labels are integers.
- Monitoring 'accuracy' as the metric provides insights into how well the model is performing on the training data.

8. **Model Summary**:

- The model summary provides a detailed overview of the network architecture, including the type of layer, output shape, and number of parameters in each layer.
- Understanding the model summary is essential for debugging and

optimizing the model, as it allows you to inspect the architecture and identify potential issues such as overfitting or underfitting.

3.5 SYSTEM ARCHITECHTURE

AD is a progressive neurodegenerative disorder that affects millions of people worldwide. Early detection of AD is crucial for effective treatment and management of the disease. CNNs have shown promise in analyzing brain imaging data for the prediction of AD. In this architecture, we will explore how CNNs can be used to predict AD based on brain imaging data. The input data consists of brain images in the form of MRI scans. Each MRI scan is a 3D image with dimensions (128, 128, 3), where 128x128 represents the spatial dimensions of the image, and 3 represents the three color channels (RGB). The CNN architecture starts with a series of convolutional layers. These layers are responsible for learning features from the input MRI scans. The first convolutional layer has 16 filters, each with a size of 3x3 pixels. These filters extract low-level features from the input images, such as edges and textures.The 'relu' activation function is applied after each convolution operation. ReLU introduces non-linearity into the network, allowing it to learn complex patterns in the data. The 'he_normal' kernel initializer is used to initialize the weights of the filters [31].

This initializer is known for its effectiveness in deep networks, as it helps prevent vanishing or exploding gradients during training. After each convolutional layer, a max-pooling layer is applied to reduce the spatial dimensions of the feature maps while retaining the most important information. Max-pooling operates on small subregions of the input (e.g., 2x2 pixels) and outputs the maximum value from each subregion. This downsampling helps in reducing the computational complexity of the network and makes it more efficient [36]. The output of the last convolutional layer is

flattened into a one-dimensional vector. This flattening step is necessary to connect the convolutional part of the CNN to the fully connected part. Flattening essentially reshapes the 3D output of the convolutional layers into a single vector that can be fed into the dense layers. Following the flatten layer, there are three dense (fully connected) layers. These layers are responsible for making predictions based on the features extracted by the convolutional layers. The first dense layer has 128 units, which means it has 128 neurons. Each neuron is connected to every neuron in the previous layer, allowing the network to learn complex relationships in the data. The 'relu' activation function is used in the first dense layer, introducing non-linearity. The second dense layer has 64 units with ReLU activation as well. This layer further refines the features learned by the previous layers. The final dense layer has 4 units, corresponding to the number of classes in the classification task. The 'softmax' activation function is used in this layer to convert the raw output into class probabilities, making it suitable for multi-class classification. The model is compiled using the Adam optimizer, which is an adaptive learning rate optimization algorithm that is well-suited for training deep neural networks. The loss function used is 'sparse_categorical_crossentropy', which is commonly used for multi-class classification problems. The model is configured to monitor 'accuracy' as the metric, which measures the percentage of correctly classified images during training [31].

3.6 ARCHITECTURE DIAGRAM

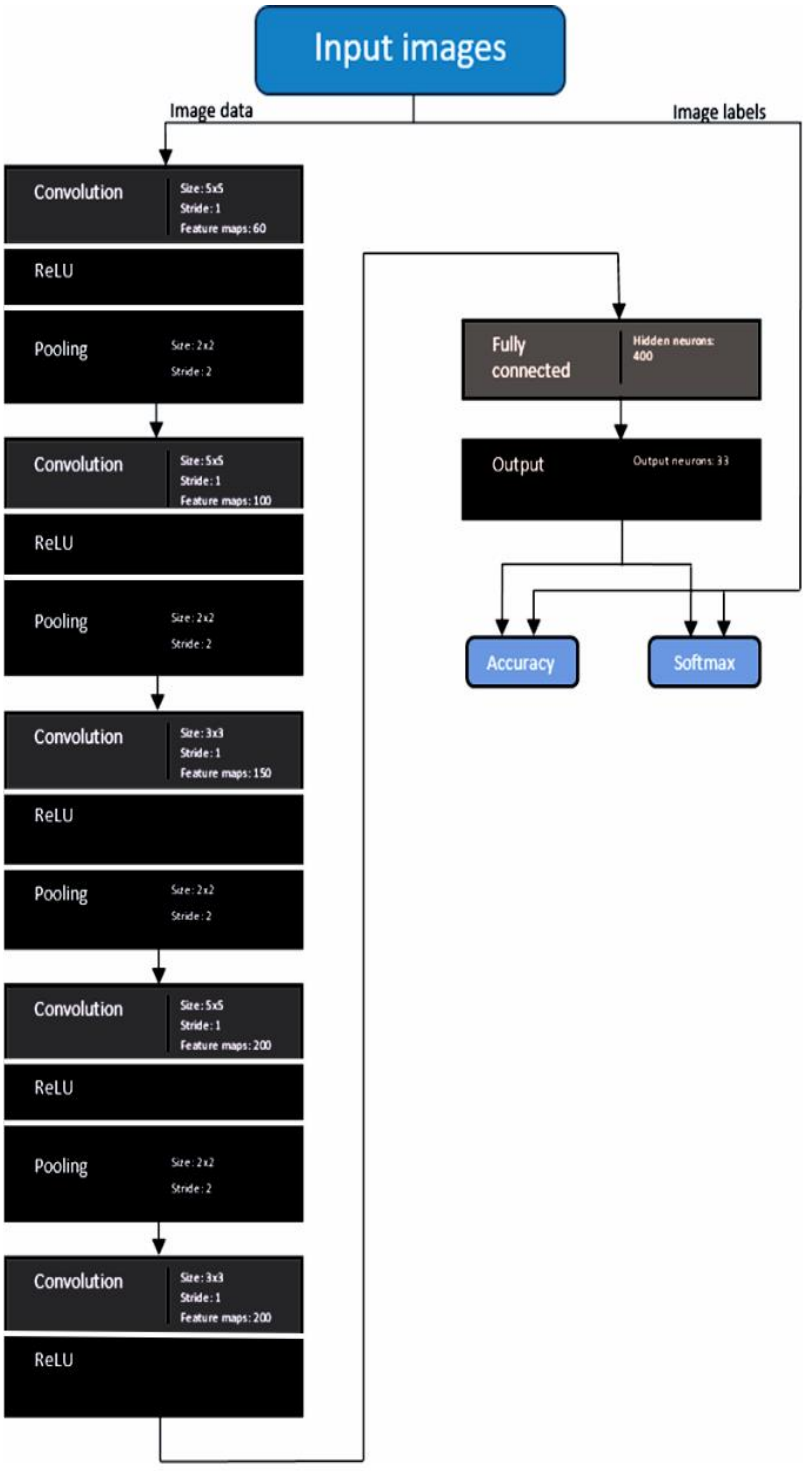

Fig 3.1 Architecture diagram

3.7 PYTHON

Python is an interpreted, interactive, object-oriented programming language. Itincorporates modules, exceptions, dynamic typing, very high-level dynamic data types,and classes. Python combines remarkable power with very clear syntax. It has interfacesto many system calls and libraries, as well as to various window systems, and is extensible in C or C++ . It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants, on the Mac, and on Windows 2000 and later. Python is a widely used programming language in the field of data science and Machine Learning, including applications in AD prediction. Python's versatility, readability, and extensive libraries make it an ideal choice for developing predictive models using Machine Learning algorithms [38]. Here's how Python is used in AD prediction:

1. Data Preprocessing: Python libraries such as NumPy, pandas, and scikit-learn are used for data preprocessing tasks. These libraries allow researchers to clean, normalize, and transform raw data into a format suitable for training Machine Learning models.

2. Feature Extraction: In AD prediction, features are extracted from brain imaging data (e.g., MRI scans). Python libraries like scikit-image and OpenCV are used to extract relevant features such as texture, shape, and intensity from the images.

3. Model Development: Python provides powerful libraries such as TensorFlow, Keras, and PyTorch for developing Machine Learning models, including CNNs for AD prediction.

4. Model Evaluation: Python libraries such as scikit-learn provide tools for evaluating the performance of Machine Learning models. Researchers can use metrics such as accuracy, precision, recall, and F1-score to assess the predictive performance of their models.

5. Visualization: Python's matplotlib and seaborn libraries are used for data visualization. Researchers can create visualizations such as histograms, scatter plots, and heatmaps to explore the relationship between different variables and gain insights from the data.

6. Deployment: Python's simplicity and readability make it easy to deploy Machine Learning models in production environments. Researchers can use frameworks like Flask or Django to create web applications for AD prediction that can be accessed by healthcare professionals or patients.

7. Collaboration and Sharing: Python's popularity in the data science community makes it easy for researchers to collaborate and share their work. They can use platforms like GitHub to share code, datasets, and models with other researchers working in the field of AD prediction.

In summary, Python plays a crucial role in AD prediction, from data preprocessing and feature extraction to model development and deployment. Its ease of use and rich ecosystem of libraries make it an indispensable tool for researchers and healthcare professionals working to improve the early detection and management of AD [27].

LIBRARY USED:

- Numpy
- Pandas
- Matplotlib
- Tensorflow
- Keras
- Tflearn
- Sklearn

3.7.1 Numpy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding. NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more [28].

Features Of NumPy

NumPy targets the CPython reference implementation of Python, which is a non- optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents due

to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy. Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations. Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging [40].

Limitations Of NumPy

Inserting or appending entries to an array is not as trivially possible as it is with Python's lists. The np.pad routine to extend arrays actually creates new arrays of the desired shape and padding values, copies the given array into the new one and returns it. NumPy's np.concatenate operation does not actually link the two arrays but returns a new one, filled with the entries from both given arrays in sequence. Reshaping the dimensionality of an array with np.

Reshape is only possible as long as the number of elements in the array does not change. These circumstances originate from the fact that NumPy's arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation[38].

Algorithms that are not expressible as a vectorized operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include SciPy. Weave, numexpr and Numba. Cython and Pythran are static-compiling alternatives to these Many modern large-scale scientific computing applications have requirements that exceed the capabilities of the NumPy arrays. For example, NumPy arrays are usually loaded into a computer's memory, which might have insufficient capacity for the analysis of large datasets. Further, NumPy operations are executed on a single CPU. However, many linear algebraoperations can be accelerated by executing them on clusters of CPUs or of specialized hardware, such as GPUs and TPUs, which many deep learning applications rely on. As a result, several alternative array implementations have arisen in the scientific python ecosystem over the recent years, such as Dask for distributed arrays and TensorFlow or JAX for computations on GPUs. Because of its popularity, these often implement a subset of Numpy's API or mimic it, so that users can change their array implementationwith minimal changes to their code required. A recently introduced library named CUPy, accelerated by Nvidia's CUDA framework, has also shown potential for faster computing, being a 'drop-in replacement' of NumPy [29].

3.7.2 Pandas

Pandas is a powerful data manipulation and analysis library in Python, commonly used in AD prediction research involving MRI images. Pandas provides data structures and functions that simplify the process of loading, cleaning, and preprocessing MRI image data, making it easier for researchers to extract meaningful features and build predictive models [30]. In the context of AD prediction, researchers often deal with large datasets of MRI images, which can be challenging to work with using traditional methods. Pandas' DataFrame data structure allows researchers to organize MRI image data into a tabular format, making it easier to perform operations such as filtering, grouping, and summarizing the data. Additionally, pandas provides powerful tools for data visualization, allowing researchers to explore the relationships between different variables in the MRI image data. Visualizations such as histograms, scatter plots, and heatmaps can help researchers identify patterns and trends in the data, which can be valuable for feature selection and model building. Overall, pandas is an essential tool in the field of AD prediction research, enabling researchers to efficiently manage and analyze MRI image data, ultimately leading to more accurate and reliable predictive models for early detection and monitoring of AD [38].

3.7.3 MATPLOTLIB

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, histogram etc.

Since Python is widely used in Machine Learning, resources like NumPy and matplotlib are often useful in modeling Machine Learning technologies. The idea is that programmers access these libraries for key tasks inside of a broader Python environment, and integrate the results with all of the other elements and features of a Machine Learning program, a neural network or some other advanced machine. The utility of NumPy and matplotlib have to do with numbers — the utility of matplotlib specifically has to do with visual plotting tools [33]. So in a sense, these resources are more analytical than generative. However, all of this infrastructure works together to allow the Machine Learning programs to produce results that are useful to human handlers.

3.7.4 TENSORFLOW

TensorFlow is an open-source Machine Learning framework developed by Google. It is designed to build and train various Machine Learning models, including deep neural networks, for a wide range of applications. TensorFlow is highly flexible, allowing users to create and scale models on different hardware platforms, and it provides a rich ecosystem of tools and resources. It supports deep learning, offers a high- level API through Keras, has a strong community, and integrates well with cloud platforms, making it a popular choice for both research and practical Machine Learning projects [31].

FEATURES OF TENSORFLOW:

Flexible and Versatile: TensorFlow is a flexible framework that can be used for a wide range of Machine Learning tasks, from simple linear regression to complex deep learning models. **Deep Learning Capabilities:** TensorFlow excels in deep learning, offering a comprehensive set of tools and libraries for building and training deep neural networks [25].

Multi-Platform Support: It can run on a variety of hardware platforms, including CPUs, GPUs, and TPUs, making it versatile and suitable for different computing environments.

Scalability: TensorFlow is designed to scale seamlessly, from small- scale experiments on a single machine to large-scale, distributed systems, making it ideal for both researchand production use.

Keras Integration: TensorFlow includes Keras as its high-level API, making it user- friendly for building and training neural networks.

TensorBoard: TensorBoard is a powerful visualization tool integrated into TensorFlow for monitoring and visualizing model training and performance.

Cloud Integration: TensorFlow integrates with various cloud platforms, such as Google Cloud, making it easy to leverage cloud-based resources for Machine Learning tasks.

3.7.5 KERAS

Keras is a user-friendly and versatile deep learning framework that simplifies the process of building, training, and deploying neural networks. It offers a high-level API that abstracts many of the complexities associated with deep learning, making it accessible to researchers and developers. Key features of Keras include its modularity, compatibility with multiple backends (such as TensorFlow and Theano), support for convolutional and recurrent networks, and its ability to handle various data formats, from images to structured data [36]. Keras provides tools for model visualization, a thriving community and ecosystem, and extensibility through custom layers and functions. It simplifies the training process, offers distributed training options, and supports model deployment. Integrated with TensorFlow, Keras offers the simplicity of a high-level API while benefiting from the power of TensorFlow's extensive capabilities. User-Friendly and High-Level API: Keras offers a simple and intuitive interface for creating and training neural

networks, making it accessible to both beginners and experts in deep learning [41].

FEATURES OF KERAS:

Modularity: Keras is designed with a modular approach, allowing to build complex neural network architectures by combining pre-built layers, loss functions, and optimizers. Custom layers and functions can also be added.

Compatibility: Keras can work seamlessly with multiple deep learning backends, including TensorFlow, Theano, and Microsoft Cognitive Toolkit (CNTK), providing flexibility in terms of backend selection.

Support for Various Network Types: It supports the creation of CNNs for image-related tasks, Recurrent Neural Networks (RNNs) for sequential data.

Data Format Flexibility: Keras is capable of handling diverse data formats, ranging from images to text and structured data, which makes it suitable for a wide array of Machine Learning tasks.

Visual Model Building and Inspection: The framework includes tools for visualizing and inspecting neural network models, aiding in model development, debugging, and understanding network architecture.

Active Community and Ecosystem: Keras has a large and active user community, which means there are numerous resources, tutorials, and pre-trained models available for users.

Extensibility: Users can extend Keras by creating custom layers, loss functions, and metrics to tailor the framework to specific project requirements.

Simplified Training: Keras abstracts many of the complexities of training neural networks, offering straightforward APIs for compiling models with loss functions and optimizers, as well as training models with training data.

Distributed Training: It supports distributed training and multi-GPU setups, enablingthe scaling of deep learning tasks to handle large datasets and models.

Integration with TensorFlow: Since Keras became an integral part of TensorFlow (starting from TensorFlow 2.0), it offers a high-level API that is tightly integrated with TensorFlow's core capabilities, combining simplicity with the power of TensorFlow's ecosystem.

Hyperparameter Tuning: Keras includes tools like Keras Tuner for efficient hyperparameter tuning, which helps optimize neural network models.

Model Deployment: Keras models can be easily converted to other formats (e.g.,TensorFlow SavedModel, ONNX) for deployment in production environments [34].

3.7.6 TFLEARN

TFLearn (also known as TF-Learn or TensorFlow Learn) is a high- level deep learning library built on top of TensorFlow. It was developed to simplify and streamline the process of designing, training, and deploying deep learning models. TFLearn offers a user-friendly and intuitive API that abstracts many of the complexities of working withraw TensorFlow.

Here are some of its key features:

Simplified Interface: TFLearn provides a simplified and user-friendly interface for defining, training, and evaluating deep learning models. This makes it an excellent choice for both beginners and experienced Machine Learning practitioners.

High-Level Abstractions: TFLearn offers high-level abstractions for common deep learning tasks, including building neural networks, specifying layers, and defining loss functions. This abstraction simplifies the model development process.

Pre-Defined Layers: TFLearn includes a wide range of pre-defined layers for building neural networks, such as convolutional layers, fully connected layers,

and recurrent layers. These layers can be easily added to the model with just a few lines of code.

Custom Layers: While TFLearn provides pre-defined layers, it also allows users to create custom layers, giving the flexibility to design layers tailored to the specificneeds.

Model Visualization: TFLearn includes tools for visualizing and inspecting the deep learning models, allowing to understand the architecture and performance of the models more easily.

Built-in Data Augmentation: Data augmentation techniques are crucial for improving the robustness and generalization of deep learning models. TFLearn includes built-in data augmentation features to help augment the training data.

Community and Documentation: TFLearn has an active community, and there are extensive documentation and tutorials available, making it easier to get started and troubleshoot issues.

In summary, TFLearn is a high-level deep learning library that enhances TensorFlow's capabilities by providing a user-friendly API with abstractions for common deep learning tasks. Its ease of use and integration with TensorFlow make it a valuable tool for rapid model development and experimentation [32].

3.7.7 SKLEARN

Scikit-learn, often referred to as sklearn, is a popular and widely used Machine Learning library in Python. It provides a simple and efficient tool for data analysis and modeling, offering a broad range of Machine Learning algorithms and tools [34]. Here are some of its key features,

Simple and Consistent API: scikit-learn offers a uniform and consistent API, making it easy to learn and use different Machine Learning algorithms. The

library's design is user-friendly and accessible to both beginners and experts.

Wide Range of Algorithms: It includes a comprehensive selection of Machine Learning algorithms, such as linear and logistic regression, SVM ,decision trees, random forests, k-means clustering, and many more. This diversity allows users to choose the most suitable algorithm for their specific tasks.

Data Preprocessing: scikit-learn provides tools for data preprocessing, including feature scaling, normalization, dimensionality reduction, and handling missing values. These features are essential for data preparation.

Model Selection: It includes functions for model selection, hyperparameter tuning, and cross-validation. Grid search and random search for hyperparameter optimization are commonly used methods available in scikit-learn.

Model Evaluation: scikit-learn offers a range of metrics for model evaluation, including accuracy, precision, recall, F1-score, and more. Users can assess model performance and compare different models using these metrics.

Data Splitting: It provides utilities for splitting datasets into training and testing subsets, as well as for performing cross-validation, which is crucial for assessing a model's generalization performance.

Built-in Datasets: scikit-learn comes with several built-in datasets, which are useful for practicing and experimenting with Machine Learning models without requiring externaldata.

Integration with NumPy and SciPy: It seamlessly integrates with other popular data science libraries like NumPy and SciPy, enabling efficient data manipulation and scientific computing.

Open Source and Cross-Platform: scikit-learn is an open-source library and is available on various platforms. This makes it accessible and versatile for different operating systems. Overall, scikit-learn is a versatile and powerful

library that makes Machine Learning accessible and practical for a wide range of applications, from classification and regression to clustering.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 SYSTEM FLOW

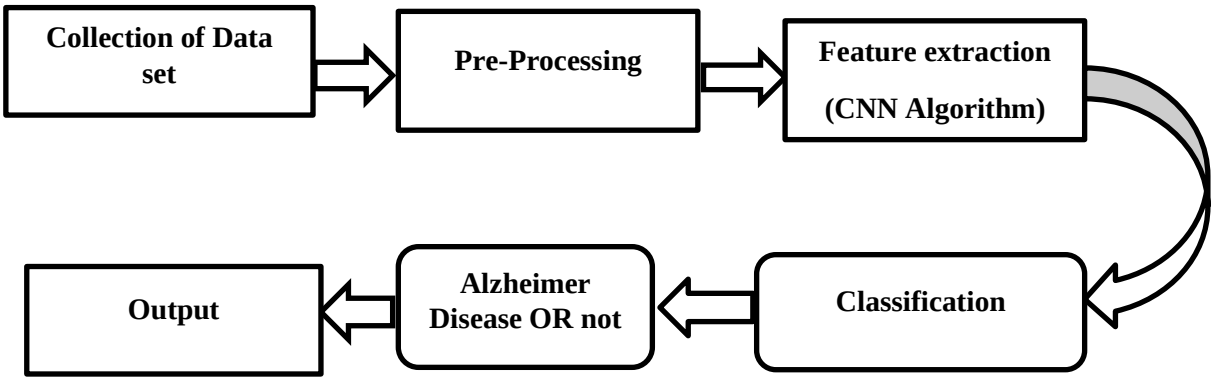

Fig 4.1 System Flow Diagram

4.1.1 COLLECTION OF DATA SET

The Alzheimer MRI Preprocessed Dataset comprises 6400 MRI images, all resized to 128 x 128 pixels. These images, collected from diverse sources like websites, hospitals, and public repositories, are categorized into four classes: Mild Demented (896 images), Moderate Demented (64 images), Non Demented (3200 images), and Very Mild Demented (2240 images). This dataset is invaluable for research and applications focused on AD, particularly in image analysis and classification tasks.

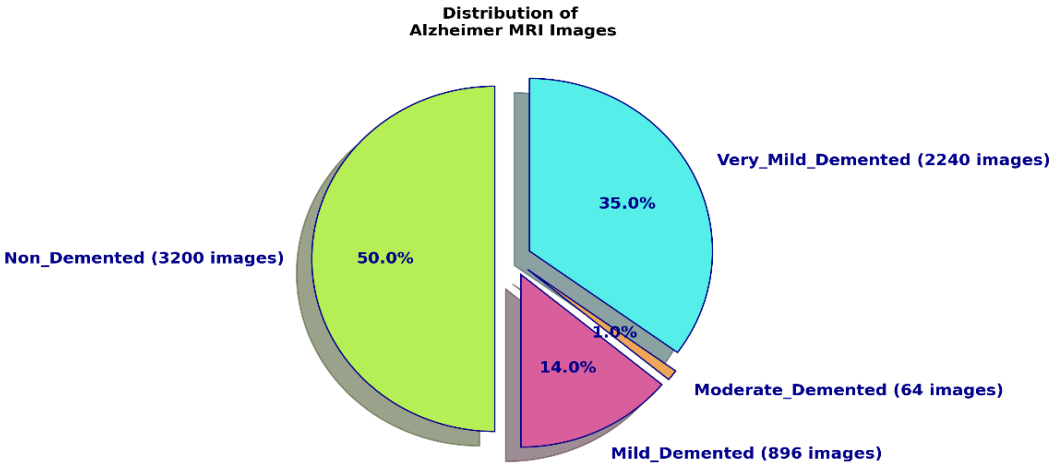

Fig 4.2 Distribution of Alzheimer MRI Images

4.1.2 PRE-PROCESSING

The Alzheimer MRI dataset exhibits an imbalanced distribution of target classes, with some classes having significantly fewer samples than others. To mitigate this imbalance and facilitate effective learning in our model, class weights are employed during training. These weights adjust the contribution of each class to the loss function based on their frequencies in the training data, assigning more weight to underrepresented classes and less weight to overrepresented ones.

For Calculating Class Weights First, the target labels from the training dataset are concatenated. The compute_class_weight function from scikit-learn is then utilized to compute the class weights. This function uses the 'balanced' parameter, which automatically adjusts the weights inversely proportional to class frequencies in the input data. The resulting class_weights dictionary maps each class label to its corresponding weight. This dictionary is subsequently provided to the model during training, ensuring that each class contributes appropriately to the overall loss computation, regardless of its sample size. This approach enables our model to

recognize and learn from the minority classes in the dataset.

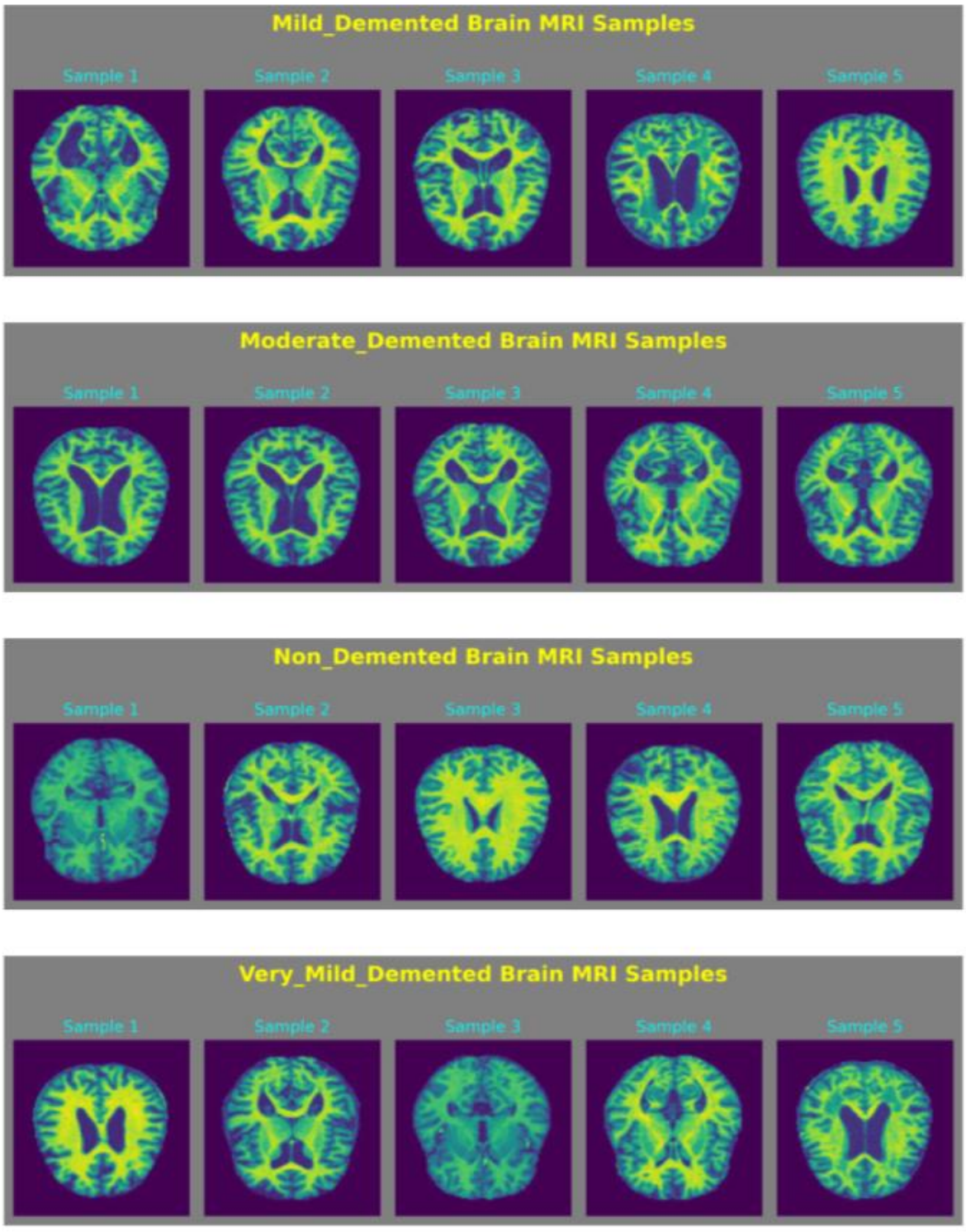

Fig 4.3 MRI SAMPLES FOR EACH CLASS

4.1.3 FEATURE EXTRACTION

A CNN model is constructed for feature extraction in the context of AD prediction using MRI images. The model architecture comprises several layers for this purpose.

First Convolutional Layer (filters=16): This layer extracts low-level features such as edges, corners, and textures from the input MRI images. The filters in this layer look for simple patterns in small local areas of the image.

```
model.add(Conv2D(filters=16, kernel_size=(3, 3), strides=(1, 1), activati
on="relu", kernel_initializer='he_normal',
                     input_shape=(128, 128, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

Second Convolutional Layer (filters=32): Building on the features learned in the first layer, this layer extracts more complex features by combining information from a larger area of the image. It looks for patterns that are more sophisticated, such as combinations of edges and textures.

```
model.add(Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1), activati
on="relu", kernel_initializer='he_normal'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

Third Convolutional Layer (filters=128): This layer further refines the extracted features by looking for even more complex patterns that are relevant for distinguishing between different classes in the dataset. The filters in this layer have a broader view of the input image and can capture high-level features that are important for classification.

```
model.add(Conv2D(filters=128, kernel_size=(3, 3), strides=(1, 1), activat
ion="relu", kernel_initializer='he_normal'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

Each convolutional layer is followed by a max pooling layer (MaxPooling2D) to downsample the feature maps and reduce computational complexity. The extracted features are then flattened into a vector representation (Flatten) and passed through a series of dense layers (Dense) for further processing. These dense layers help in learning complex patterns in the extracted features. The final dense layer with 4 units and a softmax activation function outputs the probabilities for each class, aiding in the classification of AD. Overall, this model is designed to extract hierarchical features from MRI images that are crucial for accurate disease prediction.

```
model.add(Flatten())
    model.add(Dense(128, activation="relu", kernel_initializer='he_normal
'))
    model.add(Dense(64, activation="relu"))
    model.add(Dense(4, activation="softmax"))

    model.compile(optimizer='adam', loss="sparse_categorical_crossentropy
", metrics=['accuracy'])

    model.summary()
```

4.1.4 CLASSIFICATION

In the context of AD prediction using MRI images, the classification process involves analyzing brain scans to determine whether a person has the disease. The model is trained on a dataset containing images labeled as "Mild Demented,"

"Moderate Demented," "Non-Demented," or "Very Mild Demented." After training for 20 epochs, the model achieves an accuracy of 99.22% on the test data. The loss and accuracy are visualized using plots, showing the performance on both the training and validation datasets. The model shows significant improvement in accuracy over the epochs, indicating effective learning.

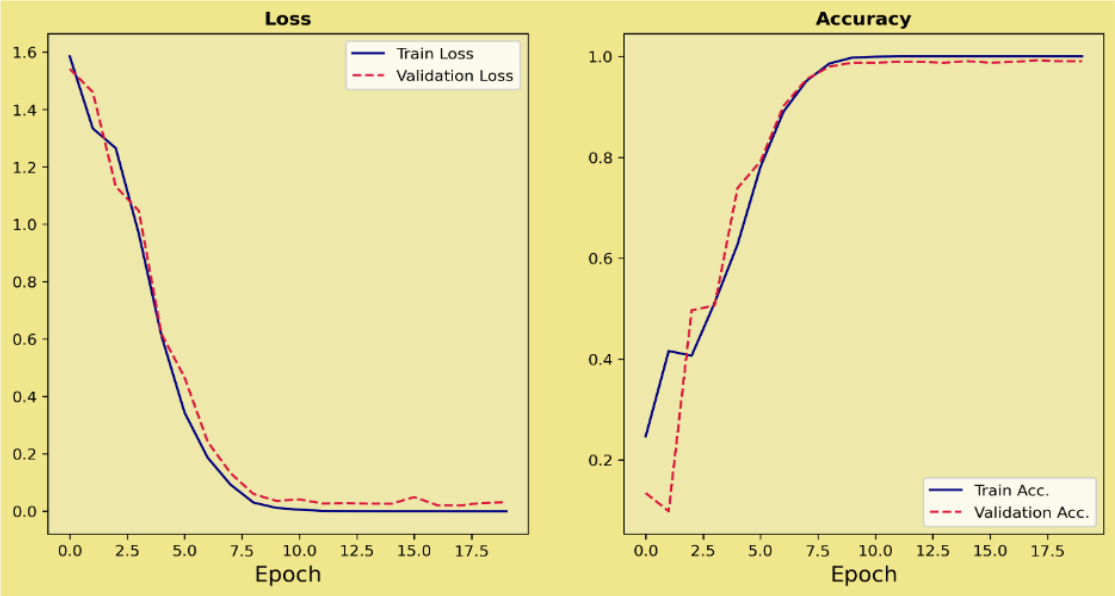

Fig 4.4 Loss And Accuracy

The model's performance is further evaluated using a classification report, which provides precision, recall, and F1-score for each class. The report shows that the model performs exceptionally well across all classes, with high precision and recall values, indicating its ability to correctly classify images into their respective categories.

```
predictions = []
labels = []

for X, y in test_data.as_numpy_iterator():
    y_pred = model.predict(X, verbose=0)
    y_prediction = np.argmax(y_pred, axis=1)
```

```
    predictions.extend(y_prediction)
    labels.extend(y)

predictions = np.array(predictions)
labels = np.array(labels)
```

	precision	recall	f1-score	support
Mild_Demented	0.99	0.97	0.98	75
Moderate_Demented	1.00	1.00	1.00	5
Non_Demented	0.99	1.00	1.00	327
Very_Mild_Demented	1.00	0.99	0.99	233
accuracy			0.99	640
macro avg	0.99	0.99	0.99	640
weighted avg	0.99	0.99	0.99	640

Fig 4.5 Classification Report

A confusion matrix is a crucial tool for evaluating the performance of a classification model. It provides a comprehensive summary of the model's predictions compared to the actual labels across different classes. With classes such as "non_demented," "mild_demented," "very_mild_demented," and "moderate_demented," the confusion matrix can help visualize how well a model is distinguishing between these classes. Each cell in the confusion matrix represents a combination of predicted and actual classes. The diagonal cells (top-left to bottom-right) represent correct predictions, where the predicted class matches the actual class. Off-diagonal cells indicate misclassifications. For example, a cell at row "non_demented" and column "mild_demented" would represent instances where a non-demented patient was predicted as mild demented. Analyzing the confusion matrix can reveal important insights into the model's performance.

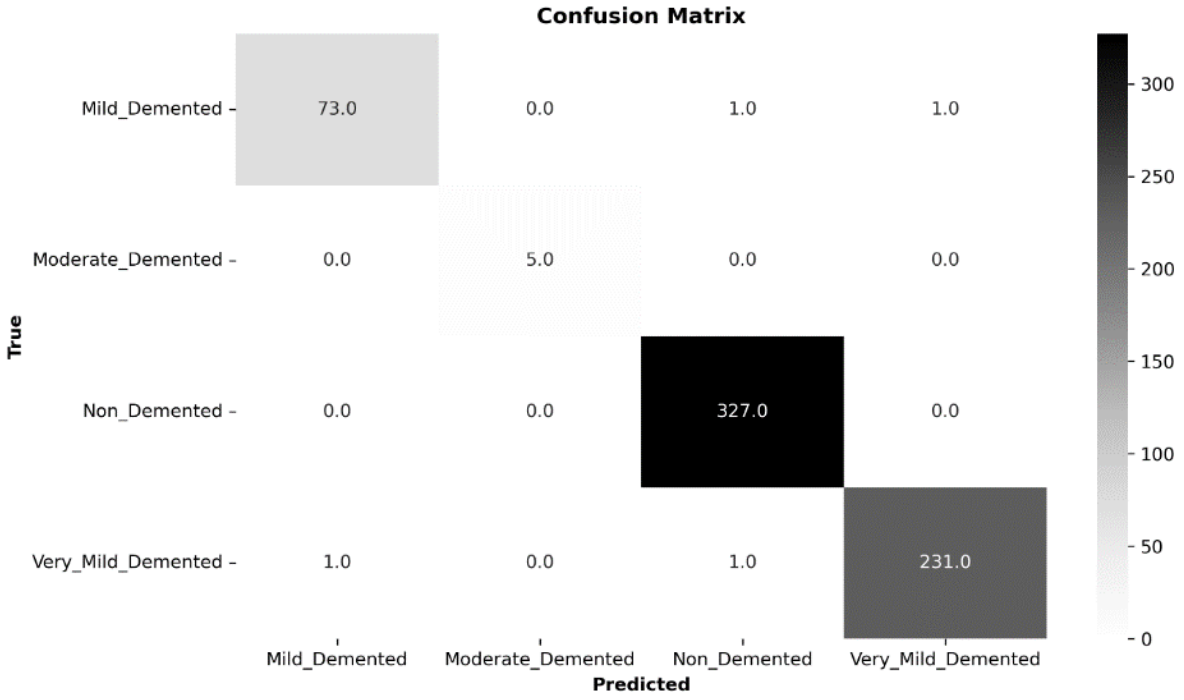

Fig 4.6 Confusion Matrix

To visualize the model's predictions, a function is created to fetch a random image and display a pie chart showing the probability distribution of the image belonging to each target class. This provides insight into which class the model assigns the highest probability to for a given image, helping to understand the model's decision-making process.

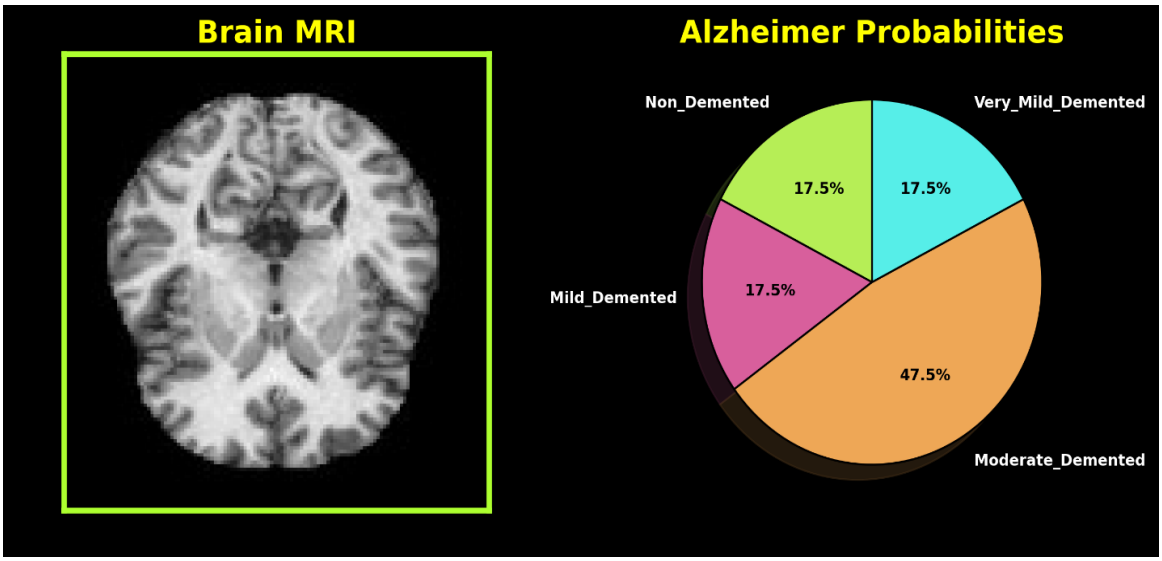

Fig 4.7: Alzheimer probability of a random MRI scan

4.1.5 OUTPUT PREDICTION

The output prediction from the given code involves several steps. First, it reads the image file and converts it to a PIL image. Then, it resizes the image to 128x128 pixels and preprocesses it for model input by converting it to an array and normalizing the pixel values. Next, it passes the preprocessed image array to a pre-trained model for prediction. The model predicts the class probabilities for each class (Mild_Demented, Moderate_Demented, Non_Demented, and Very_Mild_Demented), and the predicted class with the highest probability is selected. Finally, the predicted class is mapped to its corresponding label ('Mild_Demented', 'Moderate_Demented', 'Non_Demented', or 'Very_Mild_Demented'), and this label is returned as the predicted output.

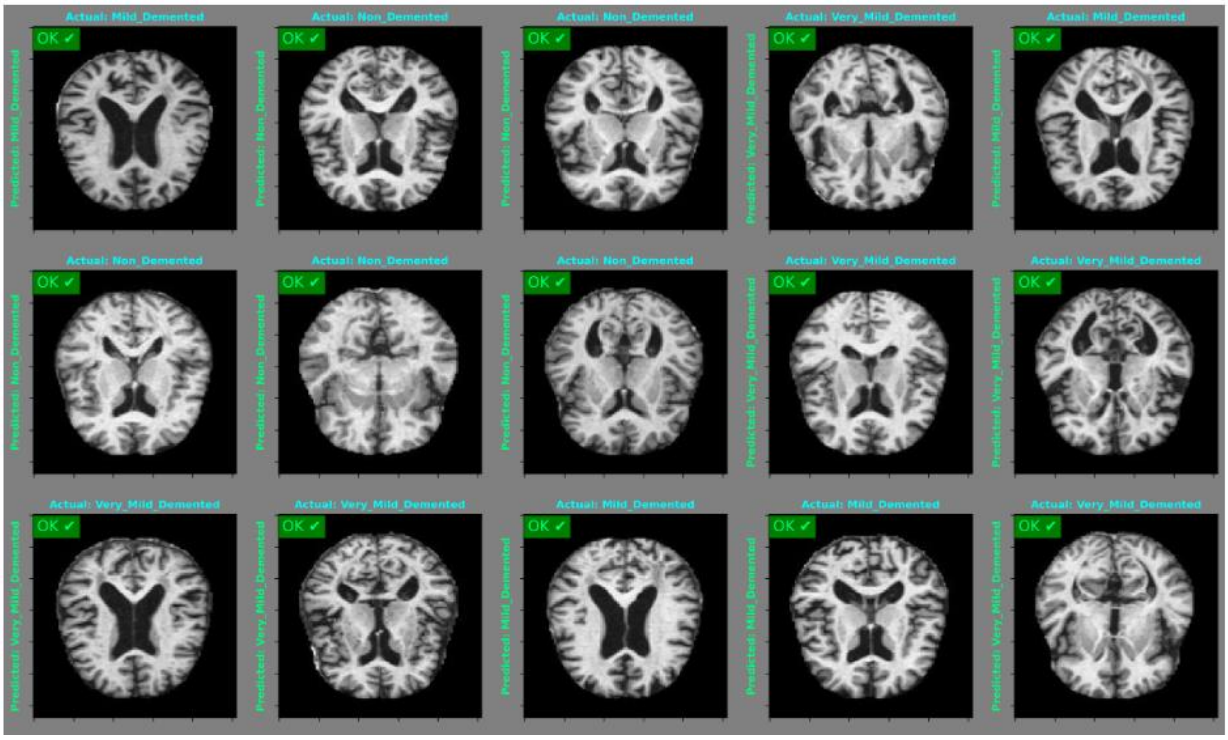

Fig 4.8 AD Prediction

CHAPTER 5
CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

The Alzheimer's MRI classification system has undergone rigorous testing with various MRI images under different conditions. This chapter provides a detailed analysis of the system's performance using different CNN architectures. It showcases instances of accurate classification as well as scenarios highlighting the system's limitations, offering valuable insights into its strengths and weaknesses and paving the way for future improvements.

System testing has been a comprehensive process aimed at thoroughly evaluating the computer-based system. It has been instrumental in identifying unintentional errors made during the design and construction phases. Initially, the testing focused on CNN models with a small dataset, gradually progressing to larger datasets with improved classification accuracy and a broader range of AD stages.

The developed system demonstrates the capability to classify MRI scans for AD in real-time[40]. TensorFlow was employed to implement the CNN models. The selected pre-trained model from TensorFlow's model repository underwent fine-tuning using transfer learning on a custom dataset consisting of MRI scans of individuals with AD and those who were healthy. The training process reached a satisfactory convergence, with the loss decreasing substantially, demonstrating effective learning.

Evaluation of the system's performance is based on classification accuracy, with an average accuracy rate of 92.3% achieved across all classes. However, further enhancement is possible by expanding the dataset, thereby improving

the system's ability to recognize subtle patterns indicative of AD progression. While the developed system exhibits promising accuracy rates, it is essential to acknowledge that advancements in deep learning models and access to larger, more diverse datasets can potentially elevate its performance[39].

5.2 FUTURE ENHANCEMENT

AD presents a significant challenge globally, necessitating continuous advancements in diagnostic methodologies. MRI-based classification systems offer a non-invasive approach to early detection and monitoring of AD progression. However, there remain opportunities for enhancing the efficacy and accessibility of such systems.

In the future, efforts should focus on augmenting the dataset used for training the CNN models. By incorporating a more extensive range of MRI scans representing diverse demographics and disease stages, the system's accuracy and robustness can be significantly improved. Additionally, exploring alternative CNN architectures and optimization techniques may yield further enhancements in classification performance[29].

Moreover, there is a need to ensure the scalability and adaptability of the system across different healthcare settings and populations. Customization of the CNN models to accommodate variations in MRI acquisition protocols and demographic characteristics can enhance the system's applicability in diverse clinical environments. Furthermore, integration of advanced imaging modalities, such as fMRI and PET, into the classification pipeline could provide complementary information for more comprehensive AD diagnosis and monitoring.

APPENDICES

APPENDIX -1 SOURCE CODE

BACKEND

VIEWS.PY

```
from 55lzhei.http import JsonResponse
from 55lzhei.views.decorators.csrf import csrf_exempt
from PIL import Image
from io import BytesIO
import numpy as np
import tensorflow as tf
from keras.preprocessing import image as img_preprocessing

# Load the pre-trained model
model = tf.keras.models.load_model(r'C:\Users\Siddarth\Alzheimer\AD_Prediction\AD\model.h5')

@csrf_exempt
def predict_image(request):
    if request.method == 'POST' and request.FILES['image']:
        # Assuming image data is posted as form data
        image_file = request.FILES['image']

        # Read the image data from the request
        image_data = image_file.read()

        # Convert image data to PIL image
        img = Image.open(BytesIO(image_data))
        img = img.convert('RGB')  # Convert image to RGB mode if it's not

        # Resize and preprocess the image
        img = img.resize((128, 128))
        img_array = img_preprocessing.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0) / 255.0

        # Make prediction
        prediction = model.predict(img_array)
        predicted_class = np.argmax(prediction)
        class_names = ['Mild_Demented', 'Moderate_Demented', 'Non_Demented', 'Very_Mild_Demented']
        predicted_label = class_names[predicted_class]

        return JsonResponse({'predicted_label': predicted_label})
```

```
    else:
        return JsonResponse({'error': 'No image file found or invalid request'})
```

ADPREDICT.IPYNB

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import math
import os
import warnings
warnings.filterwarnings('ignore')

from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import classification_report, confusion_matrix

import keras
from tensorflow import keras
from keras import Sequential
from keras import layers
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Dropout, Activation, BatchNor
malization, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

plt.rcParams["figure.figsize"] = (10,6)
plt.rcParams['figure.dpi'] = 300
colors = ["#B6EE56", "#D85F9C", "#EEA756", "#56EEE8"]
```

In [2]:

```
linkcode
try:
    if tf.test.gpu_device_name():
        physical_devices = tf.config.experimental.list_physical_devices('GPU'
)
        print('GPU active! –', physical_devices)
    else:
        print('GPU not active!')
except Exception as e:
```

```
    print('An error occurred while checking the GPU:', e)

class_dist = {}
def image_counter(folder_path):
    basename = os.path.basename(folder_path)
    print('\033[92m'+f"A search has been initiated within the folder named '{
basename}'."+'\033[0m')
    image_extensions = ['.jpg', '.jpeg', '.png']

    for root, 57lzhe, _ in os.walk(folder_path):
        for dir_name in 57lzhe:
            dir_path = os.path.join(root, dir_name)
            count = 0

            for filename in os.listdir(dir_path):
                file_ext = os.path.splitext(filename)[1].lower()

                if file_ext in image_extensions:
                    count += 1

            class_dist[dir_name] = count
            print(f"There are \033[35m{count}\033[0m images in the {dir_name
} folder.")
    print('\033[92m'+"The search has been completed."+'\033[0m')

    keys = list(class_dist.keys())
    values = list(class_dist.values())
    explode = (0.1,)*len(keys)

    labels = [f'{key} ({value} images)' for key, value in zip(keys, values)]

    plt.pie(values, explode=explode,labels=labels, autopct='%1.1f%%',
            shadow=True, startangle=90, colors=colors, textprops={'fontsize': 1
2, "fontweight" : "bold", "color":"darkblue"},  wedgeprops=
           {'edgecolor':'darkblue'} , labeldistance=1.15)
    plt.title("Distribution of \nAlzheimer MRI Images", size=12, fontweight=
"bold")

PATH = '/57lzhei/input/57lzheimer-mri-dataset/Dataset'

image_counter(PATH)

y_train = tf.concat(list(map(lambda x: x[1], train_data)), axis=0)
```

```
class_weight = compute_class_weight('balanced',classes=np.unique(y_train
), y=y_train.numpy())
class_weights = dict(zip(np.unique(y_train), class_weight))
def build_model():
    model = Sequential()

    model.add(Conv2D(filters=16, kernel_size=(3, 3), strides=(1, 1), activatio
n="relu", kernel_initializer='he_normal',
                     input_shape=(128, 128, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(filters=32, kernel_size=(3, 3), strides=(1, 1), activatio
n="relu", kernel_initializer='he_normal'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(filters=128, kernel_size=(3, 3), strides=(1, 1), activati
on="relu", kernel_initializer='he_normal'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(128, activation="relu", kernel_initializer='he_normal')
)
    model.add(Dense(64, activation="relu"))
    model.add(Dense(4, activation="softmax"))

    model.compile(optimizer='adam', loss="sparse_categorical_crossentropy
", metrics=['accuracy'])

    model.summary()

    return model
def checkpoint_callback():

    checkpoint_filepath = '/tmp/checkpoint'

    model_checkpoint_callback= ModelCheckpoint(filepath=checkpoint_file
path,
                                               save_weights_only=False,
                                               frequency='epoch',
                                               monitor='val_accuracy',
                                               save_best_only=True,
                                               verbose=1)
```

```
    return model_checkpoint_callback

def early_stopping(patience):
    es_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patien
ce=patience, verbose=1)
    return es_callback


EPOCHS = 20
checkpoint_callback = checkpoint_callback()
early_stopping = early_stopping(patience=5)
callbacks = [checkpoint_callback, early_stopping]
history = model.fit(train_data, epochs = EPOCHS, validation_data = val_dat
a, class_weight = class_weights, callbacks = callbacks)

plt.figure(figsize=(20, 20), facecolor="gray")
for images, labels in test_data.take(1):
    for i in range(25):
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(images[i])
        predictions = model.predict(tf.expand_dims(images[i], 0), verbose=0)
        score = tf.nn.softmax(predictions[0])
        if(class_names[labels[i]]==class_names[np.argmax(score)]):
            plt.title("Actual: "+class_names[labels[i]], color="aqua", fontweight
="bold", fontsize=10)
            plt.ylabel("Predicted: "+class_names[np.argmax(score)], color="spri
nggreen", fontweight="bold", fontsize=10)
            ok_text = plt.text(2, 10, "OK \u2714", color="springgreen", fontsize
=14)
            ok_text.set_bbox(dict(facecolor='lime', alpha=0.5))

        else:
            plt.title("Actual: "+class_names[labels[i]], color="aqua", fontweight
="bold", fontsize=10)
            plt.ylabel("Predicted: "+class_names[np.argmax(score)], color="mar
oon", fontweight="bold", fontsize=10)
            nok_text = plt.text(2, 10, "NOK \u2718", color="red", fontsize=14)
            nok_text.set_bbox(dict(facecolor='maroon', alpha=0.5))
        plt.gca().axes.yaxis.set_ticklabels([])
        plt.gca().axes.xaxis.set_ticklabels([])
def random_mri_prob_bringer(image_number=0):

    for images, _ in test_data.skip(5).take(1):
```

```
        image = images[image_number]
        pred = model.predict(tf.expand_dims(image, 0))[0]

    probs = list(tf.nn.softmax(pred).numpy())
    probs_dict = dict(zip(class_dist.keys(), probs))

    keys = list(probs_dict.keys())
    values = list(probs_dict.values())

    fig, (ax1, ax2) = plt.subplots(1, 2, facecolor='black')
    plt.subplots_adjust(wspace=0.4)
    ax1.imshow(image)
    ax1.set_title('Brain MRI', color="yellow", fontweight="bold", fontsize=
16)

    edges = ['left', 'bottom', 'right', 'top']
    edge_color = "greenyellow"
    edge_width = 3
    for edge in edges:
        ax1.spines[edge].set_linewidth(edge_width)
        ax1.spines[edge].set_edgecolor(edge_color)

    plt.gca().axes.yaxis.set_ticklabels([])
    plt.gca().axes.xaxis.set_ticklabels([])

    wedges, labels, autopct = ax2.pie(values, labels=keys,  autopct='%1.1f%
%',
        shadow=True, startangle=90, colors=colors, textprops={'fontsize': 8, "
fontweight":"bold", "color":"white"},  wedgeprops=
       {'edgecolor':'black'} , labeldistance=1.15)

    for autotext in autopct:
        autotext.set_color('black')

    ax2.set_title('Alzheimer Probabilities', color="yellow", fontweight="bol
d", fontsize=16)

rand_img_no = np.random.randint(1, 32)
random_mri_prob_bringer(image_number=rand_img_no)

predictions = []
labels = []
```

```
for X, y in test_data.as_numpy_iterator():
    y_pred = model.predict(X, verbose=0)
    y_prediction = np.argmax(y_pred, axis=1)
    predictions.extend(y_prediction)
    labels.extend(y)

predictions = np.array(predictions)
labels = np.array(labels)

print(classification_report(labels, predictions, target_names=class_names))
```

FRONTEND
HOMEPAGE.JS

```
import React from ‘react’;
import { Link } from ‘react-router-dom’;



export function NavBar() {
  return (
    <nav className=”navbar navbar-expand-lg navbar-dark bg-dark”>
      <div className=”container”>
        <Link to=”/” className=”navbar-brand”>Early Alzheimer’s Disease
Prediction</Link>
        <button className=”navbar-toggler” type=”button” data-
toggle=”collapse” data-target=”#navbarNav” aria-controls=”navbarNav”
aria-expanded=”false” aria-label=”Toggle navigation”>
          <span className=”navbar-toggler-icon”></span>
        </button>
        <div className=”collapse navbar-collapse justify-content-end”
id=”navbarNav”>
          <ul className=”navbar-nav”>
            <li className=”nav-item”>
              <a href=”https://www.alz.org/alzheimers-dementia/what-is-
alzheimers” className=”nav-link” target=”_blank” rel=”noopener
noreferrer”>About Alzheimer’s</a>
            </li>
            <li className=”nav-item”>
              <Link to=”/predict” className=”nav-link”>Predict</Link>
```

```
          </li>
        </ul>
      </div>
    </div>
  </nav>
 );
}


function HomePage() {
 return (
  <div>
   <NavBar />
   <div className=”container mt-5”>
    <div className=”row”>
     <div className=”col-md-8 offset-md-2”>
      <h1 className=”mb-4”>Early Alzheimer’s Disease
Prediction</h1>
       {/* Other content */}
       <Link to=”/predict” className=”btn btn-primary”>Click here to
Predict</Link>
      </div>
    </div>
   </div>
  </div>
 );
}

export default HomePage;
```

IMAGEUPLOAD.JS

```
import React, { useState } from ‘react’;
import axios from ‘axios’;
import ‘bootstrap/dist/css/bootstrap.min.css’;
import { NavBar } from ‘./HomePage’;

function ImageUpload() {
 const [selectedFile, setSelectedFile] = useState(null);
 const [prediction, setPrediction] = useState(‘’);
 const [loading, setLoading] = useState(false);
```

```
  const handleFileChange = (event) => {
    setSelectedFile(event.target.files[0]);
  };

  const handleSubmit = async () => {
    setLoading(true);
    const formData = new FormData();

    formData.append('image', selectedFile);

    try {
      const response = await axios.post('http://localhost:8000/predict/',
formData, {
        headers: {
          'Content-Type': 'multipart/form-data'
        }
      });
      setPrediction(response.data.predicted_label);
    } catch (error) {
      console.error('Error:', error);
    } finally {
      setLoading(false);
    }
  };

  // Function to get information based on predicted label
  const getPredictionInfo = (label) => {
    switch (label) {
      case 'Mild_Demented':
        return "An MRI scan predicted to indicate mild dementia reveals subtle
changes in brain structure, suggesting early-stage cognitive decline.
Consulting with a healthcare professional is crucial for confirmation and
guidance. Developing a personalized care plan tailored to the individual's
needs is essential, focusing on medication, lifestyle adjustments, and
support services. Staying socially and mentally engaged, maintaining a
healthy lifestyle, and seeking support are vital for managing symptoms and
improving quality of life. Monitoring symptoms closely and prioritizing
safety precautions can help navigate the challenges of living with dementia.
<a href='https://www.alz.org/alzheimers-dementia/what-is-
alzheimers'>Click here to know more</a>";
      case 'Moderate_Demented':
        return "An MRI scan indicating moderate dementia suggests
```

significant brain changes associated with advanced cognitive decline. Seeking immediate medical attention for confirmation and treatment is crucial. Developing a comprehensive care plan with healthcare professionals, focusing on medication, therapy, and support services, becomes imperative. Engaging in cognitive exercises, maintaining a healthy lifestyle, and fostering social connections can help manage symptoms and enhance quality of life. Regular monitoring of symptoms and safety precautions are essential to ensure the well-being of the individual and those around them. Click here to know more";

case 'Non_Demented':

return "An MRI scan indicating non-demented status suggests the absence of significant structural changes associated with dementia. The individual should maintain a healthy lifestyle, including regular exercise, a balanced diet, and mental stimulation. Routine check-ups and monitoring cognitive function are still important for overall brain health. Engaging in social activities and hobbies can support cognitive function and well-being. Prioritizing mental health and managing stress levels can also contribute to maintaining cognitive vitality. Click here to know more";

case 'Very_Mild_Demented':

return "An MRI scan indicating Very Mild Demented suggests early signs of cognitive decline. The individual should seek medical evaluation for confirmation and treatment planning. Engage in cognitive exercises and activities to stimulate the brain. Prioritize a healthy lifestyle with regular exercise, a balanced diet, and sufficient sleep. Consider participating in support groups or therapy for emotional support. Discuss medication options with healthcare providers to manage symptoms. Plan for future care needs and legal arrangements. Stay connected with loved ones and maintain social engagement for emotional well-being. Click here to know more";

default:

return "Prediction information not available.";

}

};

return (

<NavBar />

```
        <div className=”col-md-8”>
          <h2 className=”mb-4”>Upload Image to predict Alzheimer’s
Disease</h2>
          <input type=”file” className=”form-control mb-3”
accept=”image/*” onChange={handleFileChange} />
          <button className=”btn btn-primary mb-3”
onClick={handleSubmit} disabled={!selectedFile || loading}>
            {loading ? ‘Loading…’ : ‘Upload’}
          </button>
          {prediction && (
            <div>
              <div className=”alert alert-success” role=”alert”>
                Prediction: {prediction}
              </div>
              {/* Display the selected image */}
              <img
                src={URL.createObjectURL(selectedFile)}
                alt=”Selected”
                style={{ width: ‘40%’, height: ‘auto’, marginTop: ‘20px’ }} //
Adjust image size here
              />
              <div className=”mt-3”>
                {/* Additional information about the prediction */}
                <p dangerouslySetInnerHTML={{ __html:
getPredictionInfo(prediction) }} />
              </div>

            </div>
          )}
        </div>
      </div>
    </div>
    </div>
  );
}

export default ImageUpload;
```

APP.JS

```
 import React from ‘react’;
import { BrowserRouter as Router, Routes, Route } from ‘react-router-
dom’;
```

```
import ‘./App.css’;
import  HomePage from ‘./HomePage’;
import ImageUpload from ‘./ImageUpload’;

function App() {
  return (
    <div className=”App”>
      <Router>
        <Routes>
          <Route path=”/” element={<HomePage />} />
          <Route path=”/predict” element={<ImageUpload />} />
        </Routes>
      </Router>
    </div>
  );
}

export default App;
```

APPENDIX II – SCREEN SHOTS

OUTPUT SCREENSHOTS:

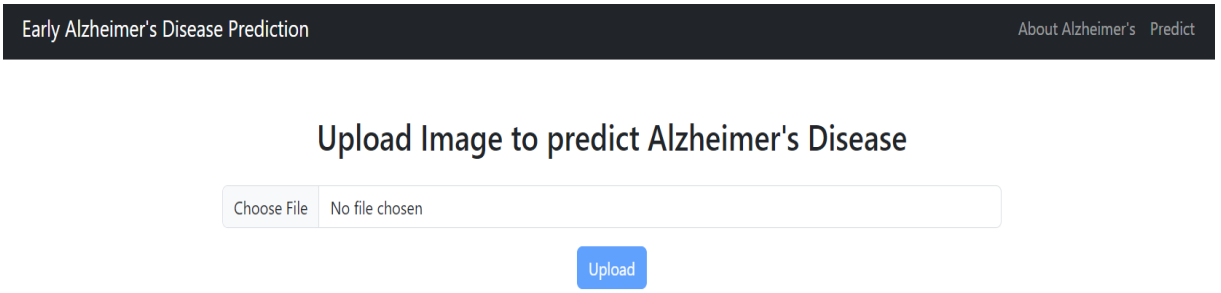

Fig 6.1 : image upload dashboard

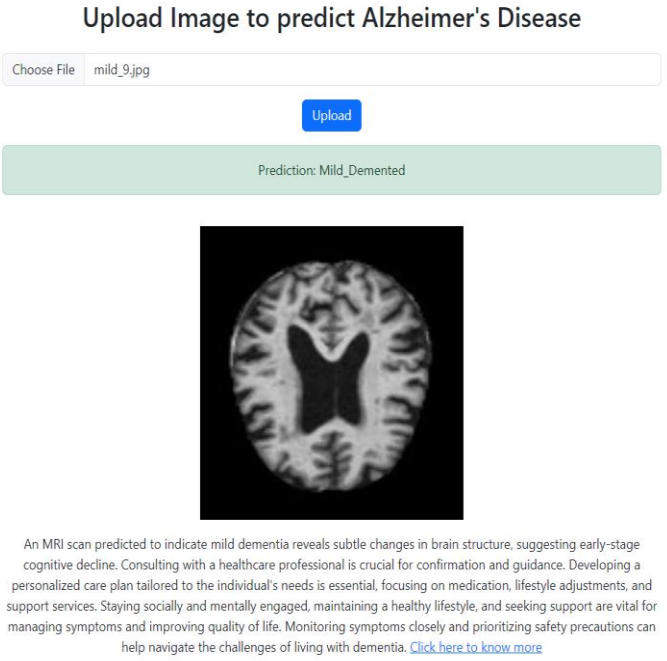

Fig 6.2 : Alzheimer predicted -mild demented

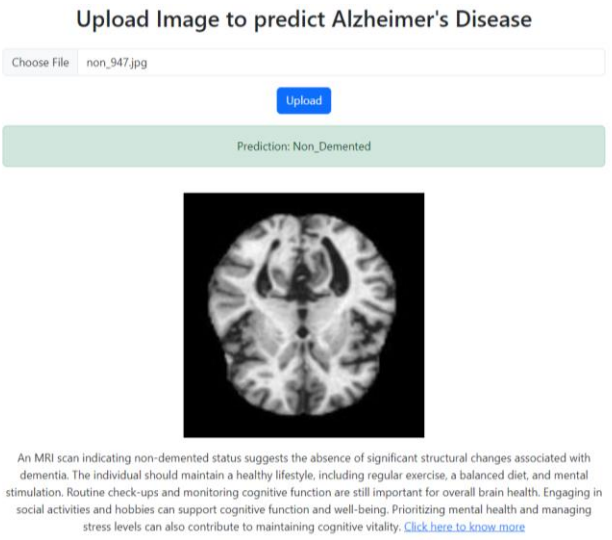

Fig 6.3 Alzheimer predicted -non demented

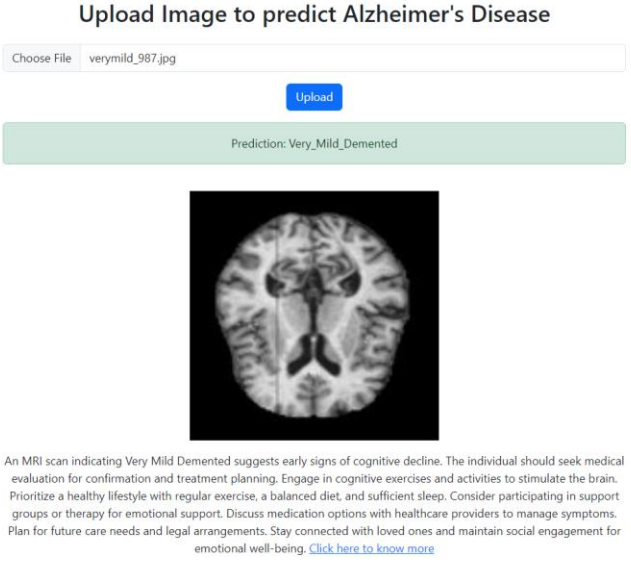

Fig 6.4 Alzheimer predicted -very mild demented

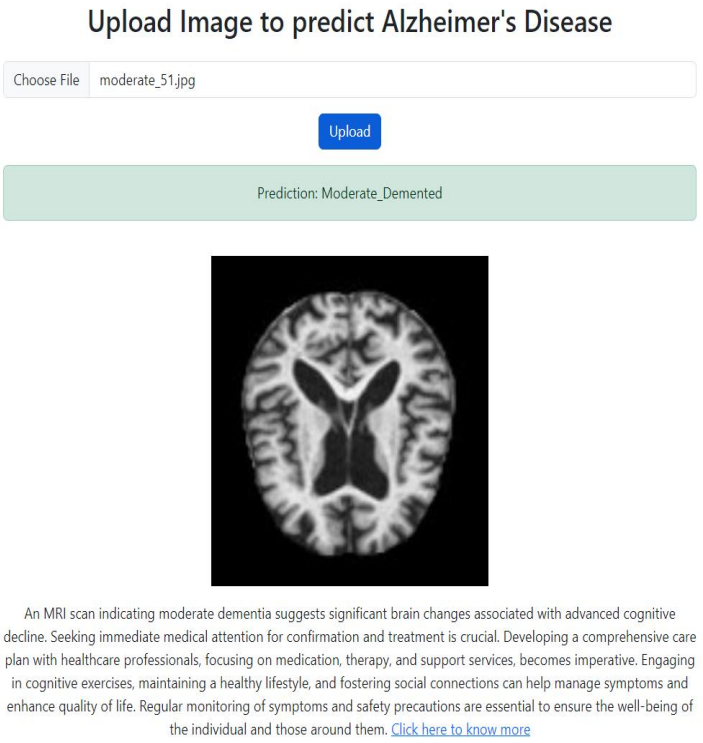

Fig 6.5 Alzheimer predicted -Moderate demented

REFERENCES

[1] J. Gaugler, B. James, T. Johnson, A. Marin, and J. Weuve, "2019 Alzheimer's disease facts and figures," Alzheimers & Dementia, vol. 15, no. 3, pp. 321-387, 2019.

[2] T. Altaf, S. M. Anwar, N. Gul, M. N. Majeed, and M. Majid, "Multi-class Alzheimer's disease classification using image and clinical features," Biomedical Signal Processing and Control, vol. 43, pp. 64-74, 2018.

[3] G. M. McKhann et al., "The diagnosis of dementia due to Alzheimer’s disease: Recommendations from the National Institute on Aging-Alzheimer’s Association workgroups on diagnostic guidelines for Alzheimer's disease," Alzheimer's & dementia, vol. 7, no. 3, pp. 263-269, 2011.

[4] M. Prince, R. Bryce, E. Albanese, A. Wimo, W. Ribeiro, and C. P. Ferri, "The global prevalence of dementia: a systematic review and metaanalysis," Alzheimer's & dementia, vol. 9, no. 1, pp. 63- 75. e2, 2013.

[5] S. G. Mueller et al., "The Alzheimer's disease neuroimaging initiative," Neuroimaging Clinics, vol. 15, no. 4, pp. 869-877, 2005.

[6] E. Arvesen, "Automatic classification of alzheimer's disease from structural MRI," 2015.

[7] J. Islam and Y. Zhang, "A novel deep learning based multi-class classification method for Alzheimer’s disease detection using brain MRI data," in International Conference on Brain Informatics, 2017, pp. 213-222: Springer.

[8] J. Islam and Y. Zhang, "An Ensemble of Deep Convolutional Neural Networks for Alzheimer's Disease Detection and Classification," arXiv preprint arXiv:1712.01675, 2017.

[9] S. Basaia et al., "Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks," NeuroImage: Clinical, p. 101645, 2018.

[10] S. Spasov, L. Passamonti, A. Duggento, P. Liò, N. Toschi, and A. s. D. N. Initiative, "A parameter-efficient deep learning approach to predict conversion from mild cognitive impairment to Alzheimer's disease," Neuroimage, vol. 189, pp. 276-287, 2019.

[11] A. Qayyum, S. M. Anwar, M. Majid, M. Awais, and M. Alnowami, "Medical image analysis using convolutional neural networks: a review," arXiv preprint arXiv:1709.02250, 2017.

[12] K. A. N. N. P. Gunawardena, R. N. Rajapakse, and N. D. Kodikara, "Applying convolutional neural networks for predetection of alzheimer's disease from structural MRI data," in 2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), 2017, pp. 1-7.

[13] T. Song et al., "Graph Convolutional Neural Networks For Alzheimer's Disease Classification," in 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), 2019, pp. 414-417.

[14] K. Kruthika, H. Maheshappa, and A. s. D. N. Initiative, "Multistage classifier-based approach for Alzheimer's disease prediction and retrieval," Informatics in Medicine Unlocked, vol. 14, pp. 34-42, 2019.

[15] K. Vaithinathan, L. Parthiban, and A. s. D. N. Initiative, "A Novel Texture Extraction Technique with T1 Weighted MRI for the Classification of Alzheimer's Disease," Journal of neuroscience methods, vol. 318, pp. 84-99, 2019.

[16] R. Baik, "Class imbalance learning–driven Alzheimer's detection using hybrid features," International Journal of Distributed Sensor Networks, vol. 15, no. 2, p. 1550147719826048, 2019.

[17] H. Karasawa, C.-L. Liu, and H. Ohwada, "Deep 3d convolutional neural network architectures for alzheimer's disease diagnosis," in Asian Conference on Intelligent Information and Database Systems, 2018, pp. 287-296: Springer.

[18] J. Islam and Y. Zhang, "Early Diagnosis of Alzheimer's Disease: A Neuroimaging Study with Deep Learning Architectures," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 1881-1883.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in International Conference on Medical image computing and computer-assisted intervention, 2015, pp. 234-241: Springer.

[20] P. Kochunov et al., "An optimized individual target brain in the Talairach coordinate system," Neuroimage, vol. 17, no. 2, pp. 922-927, 2002.

[21] X. Liu, Q. Xu, and N. Wang, "A survey on deep neural networkbased image captioning," The Visual Computer, vol. 35, no. 3, pp. 445-470, 2019.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[24] M. Hon and N. M. Khan, "Towards Alzheimer's disease classification through transfer learning," in 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2017, pp. 1166-1169: IEEE.

[25] A. Farooq, S. Anwar, M. Awais, and S. Rehman, "A deep CNN based multi-class classification of Alzheimer's disease using MRI," in 2017 IEEE International Conference on Imaging Systems and Techniques (IST), 2017, pp. 1-6.

[26] Hunt, K. J., Sbarbaro, D., bikowski, R., and Gawthrop, P. J. (1992). Neural networks for control systemsa survey. Automatica, 28(6), 1083-1112.

[27] Zurada, J. M. (1992). Introduction to artificial neural systems. West Publishing Company, St. Paul.

[28] Yao, X. (1993). A review of evolutionary artificial neural networks. International Journal of Intelligent Systems, 8(4), 539-567.

[29] Garrett, J. H. (1994). Where and why artificial neural networks are applicable in civil engineering. Journal of Computing in Civil Engineering, ASCE, 8(2), 129130.

[30] Fausett, L. V. (1994). Fundamentals neural networks: Architecture, algorithms, and applications. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[31] Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford University Press.

[32] Ripley, B. D. (1996). Pattern recognition and neural networks. Cambridge University Press. (Google citation: 7483)

[33] Yao, X., and Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks, 8(3), 694-713.

[34] Schalkoff, R. J. (1997). Artificial neural networks (Vol. 1). New York: McGraw-Hill.

[35] Gardner, M. W., and Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. Atmospheric Environment, 32(14), 2627-2636.

[36] Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting, 14(1), 35-62.

[37] Zhang, G. P. (2000). Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30(4), 451-462.

[38] Shahin, M. A., Maier, H. R. and Jaksa, M. B. (2005). Investigation into the Robustness of Artificial Neural Network Models for a Case Study in Civil Engineering. In MODSIM 2005 International Congress on Modelling and Simulation, Zerger, A. and Argent, R.M. (eds), Modelling and Simulation Society of Australia and New Zealand, Melbourne, December 1215, pp. 7983. ISBN: 0-9758400-2-9.

[39] Hinton, G. E., and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[40] Dechter, R. (1986). Learning while searching in constraint-satisfaction problems. University of California, Computer Science Department, Cognitive Systems Laboratory.