AUTOMATED DETECTION OF ALZHEMER DISEASE USING CNN ALGORITHM

A PROJECT REPORT

Submitted by

DINESH KUMAR V [REGISTER NO: 2114190104070]
DIWAKAR V [REGISTER NO: 211419104073]
DEVENTHIRAN R [REGISTER NO: 211419104055]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"AUTOMATED DETECTION OF ALZHEMER DISEASE USING CNN ALGORITHM"** is the bonafide work of **" DINESH KUMAR V (211419104070), DIWAKAR V(211419104073), DEVENTHIRAN R(211419104055)** " who carried out the project work under mysupervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA,M.E.,Ph.D., **HEAD OF THE DEPARTMENT**	**Mrs.SANGEETHA M.E.,(Ph.D),** **SUPERVISOR** **ASSISTANT PROFESSOR**
DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, POONAMALLEE, CHENNAI-600 123.	DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, POONAMALLEE, CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on..........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **DINESHKUMAR V (211419104070),DIWAKAR V(211419104073), DEVENTHIRAN R(211419104055)** hereby declare that this project report titled **"AUTOMATED DETECTION OF ALZHEMER DISEASE USING CNN ALGORITHM ",** under the guidance of **Mrs.SANGEETHA M.E .,(Ph.D.)**, is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

DINESH KUMAR V
DIWAKAR V
DEVENTHIRAN R

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank our parents, friends, project Guide **Mrs.SANGEETHA M.E.,(Ph.D.)**, and coordinator **Dr.N.PUGHAZENDI M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

DINESH KUMAR V
DIWAKAR V
DEVENTHIRAN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS

AI	Artificial Intelligence
YOLO	You Only Look Once
OCR	Optical Character Recognition
ROI	Region Of Interest
CNN	Convolutional Neural Network
OpenCV	Open Source Computer Vision
RCNN	Regions Convolutional Neural Networks
RTO	Road Traffic Officer
UML	Unified Modeling Language
FPS	Frame Per Second
GPU	Graphical Processing Unit
CCTV	Closed-Circuit Television

ABSTRACT

Many different architectures of Convolutional Neural Networks (CNNs) have been developed for use in classifying images and recognizing objects. When it comes to image-based categorization, handling hundreds of MRI image slices that are essentially identical across patients is a challenging task for CNN. By utilizing a 2D CNN design, it becomes difficult to confidently categorize a large number of individuals as having Alzheimer's disease, mild cognitive impairment, or normal cognition. To solve this problem, we have streamlined the concept of patient classification based on 3D MRI while still giving due credit to the 2D features derived from the CNN framework. Here, we share our approach to extracting 2D features from MRI scans in a format that can be used in a classification system. Our experiment demonstrates the outcome of categorizing 3 patient participants into 2 groups. After reducing the dimensionality of a 2D image with principal component analysis and truncated sparse encoding (PCA+TSNE), we used a convolutional neural network (CNN) to extract generic features for classification. Despite the lackluster performance, this seems to be an improvement over probability-based categorization using a CNN that was trained from scratch. The created feature is highly malleable and can be fine-tuned to improve precision, responsiveness, and specificity.

CHAPTER 1

INTRODUCATION

1. INTRODUCTION

1.1 OVERVIEW

There has been a lot of work done to process, simulate, and interpret the results of medical examinations using various imaging techniques such as MRI (Magnetic Imaging Resonance), PET (Positronemission tomography), and Computed Tomography (CT) scans. This is being done for the purpose of Computer Aided Diagnosis (CAD), which will be of vital importance to medical professionals. In a similar vein, multiple researchers have conducted a variety of studies using MRI as the primary biomarker in order to effectively construct a Computer Assisted Diagnosis (CAD) system for the diagnosis and detection of Alzheimer's disease.

Alzheimer's disease (AD), Parkinson's disease (PD), Huntington's disease (HD), and other neurodegenerative disorders are all included under the umbrella term neuro-degenerative diseases, which mostly affect the older population. For example, Alzheimer's disease is one of the most frequent types of dementia. Several of the symptoms of Alzheimer's disease are well-known, including problems sleeping, memory loss, and dramatic shifts in mood. According to research that was published by the World Health Organization (WHO), there are roughly 40 million people around the world who are afflicted with AD.

It is anticipated that this number will reach 135 million by the year 2050. In spite of the fact that Alzheimer's disease cannot be cured, early diagnosis and appropriate treatment of the condition can slow down the rate at which neurons are lost. This is an essential aspect of the disease that should not be overlooked. According to the data presented in the WHO report, there is an urgent requirement for the research and development of computer-aided diagnostics systems for the prompt diagnosis of AD. This encourages the researchers to direct their attention more intently towards the creation of cutting-edge diagnostic tools and algorithms for the early detection and classification of Alzheimer's disease.

Based on statistical movements of the MR image intensities, demonstrated a method for distinguishing healthy controls from people with Alzheimer's disease (AD). For the purpose of picture classification, fuzzy logic is utilized as an algorithmic classification method. In the beginning, the regions of interest were given to the fuzzy interference system as an input function. These regions were determined based on the statistical motions of the MR images, which had 24 different intensities. When calculating the area under the curve, the k-fold cross validation method is the one that is utilized.

The system's performance efficiency can be represented by this AUC calculation that was made. In addition to this, the maximum number of feature shifts that are necessary to achieve optimal AUC was studied and mentioned. Outlined the methods and advantages of the Gaussians Mixture Model, which is utilized for the classification of images to diagnose Alzheimer's disease (AD). Images of the diffusion tensor are given a classification using the GMM in order to locate the AD. In the work that is being proposed, GMM is used in conjunction with functional anisotropy (FA) and mean diffusivity (MD) in order to conduct an analysis.

Comparisons are made between the GMM and the linear discriminant performance analyses. According to the findings, the generalized estimating equation (GMM) provides a higher level of accuracy than the linear discriminant technique.

1.2 PROBLEM SOLUTION

There is no proper awareness about Alzheimer Disease. As they age, they may experience changes in your physical abilities and walking, sitting, and eventually swallowing. Individuals may need substantial assistance with daily activities as their memory, and cognitive skills continue to decline. At this stage, individuals may need 24/7 assistance for personal care and daily activities. When people suffer from dementia, their ability to communicate, adapt to their environment, and eventually move is lost. It becomes much more difficult for them to communicate pain through words or phrases.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Ensembles of Patch-Based Classifiers for Alzheimer Diseases

**Author Name : **Samsuddin Ahmed, Kyu Yeong Choi

**Year of Publish : **2019

Traditional machine learning methods are being used to automatically diagnose Alzheimer's disease (AD), while deep learning-based methods are becoming prominent. State-of-the-art multimodal diagnosis methods outperform manual diagnosis. Unfortunately, gathering data from diverse modalities is time-consuming, expensive, and may have radioactive adverse effects. Structural magnetic resonance imaging is our focus (sMRI). Our goal: 1) to boost accuracy to match state-of-the-art approaches, 2) to overcome the overfitting problem, and 3) to assess brain landmarks that yield AD diagnosis features. We targeted the left and right hippocampuses here. First, we use ensembles of simple convolutional neural networks (CNNs) as feature extractors and softmax cross-entropy as the classifier. Given data paucity, we used a patch-based strategy. Gwangju's National Research Center for Dementia (GARD) cohort dataset was used for our project. We manually located the left and right hippocampus and supplied the CNN three view patches (TVPs) after preprocessing. 90.05% accuracy. Our model performed comparably to state-of-the-art approaches on the same dataset.

2.2 Task-Induced Pyramid and Attention GAN for Multimodal Brain Image Imputation and Classification in Alzheimer's Disease

**Author Name : **Xingyu Gao, Feng Shi

**Year of Publish : **2022

Medical imaging technologies like MRI and PET can detect subtle brain anatomical and functional changes, making brain disorders like Alzheimer's disease easier to diagnose (AD). As PET is expensive or unavailable, multimodal pictures may be incomplete. Most approaches removed missing data, reducing the sample size. Multimodal feature extraction and combination remain difficult. We present a deep learning system for multimodal brain image imputation and classification using a task-induced pyramid and attention generative adversarial network (TPA-GAN) and a pathwise transfer dense convolution network (PT-DCN). First, we present a TPA-GAN with pyramid convolution, attention module, and illness classification task to generate missing PET data from MRI. Using imputed multimodal pictures, we design a dense convolution network with pathwise transfer blocks to gradually learn and incorporate multimodal features for disease categorization. Our technique outperforms state-of-the-art image imputation and brain illness diagnosis using ADNI-1/2 datasets.

2.3 A Multi-Stream Convolutional Neural Network for Classification of Progressive MCI in Alzheimer's Disease Using Structural MRI Images

**Author Name : ** Mona Ashtari-Majlan, Abbas Seifi

**Year of Publish : ** 2022

Although some individuals with progressing MCI will become Alzheimer's disease, early detection of Alzheimer's disease and its prodromal stage, also known as mild cognitive impairment (MCI), is crucial. For the purpose of differentiating between static MCI and progressive MCI, we present a multi-stream deep convolutional neural network fed with patch-based imaging data. To begin, we use a multivariate statistical test to compare MRI scans of people with Alzheimer's disease and cognitively normal patients in order to find different anatomical landmarks. To identify MRI scans, the proposed multi-stream convolutional neural network is supplied with extracted patches based on these landmarks. To make up for the dearth of progressive MCI training data, we next train the architecture in a distinct scenario utilising samples from Alzheimer's disease images, which are physically comparable to the ones of progressive MCI, and cognitively normal images. Lastly, we fine-tune the model using progressive MCI and stable MCI data by transferring the trained model weights to the suggested architecture. Based on experimental results using the ADNI-1 dataset, we may conclude that our method achieves a higher F1-score (85.96%) than competing methods for classifying patients with MCI.

2.4 Deep Learning Based Binary Classification for Alzheimer's Disease Detection using Brain MRI Images

**Author Name : ** Emtiaz Hussain, Mahmudul Hasan

**Year of Publish : ** 2020

Alzheimer's disease is an irremediable, ongoing brain ailment that gradually damages memory and thinking skills and, finally, the ability to carry out the simplest tasks. Globally, it's a major disease. Alzheimer's is incurable. Machine learning techniques, especially deep learning-based Convolutional Neural Network (CNN), are utilised to improve the procedure for the detection of Alzheimer's disease. CNN has excelled in MRI image analysis and biomedical research recently. CNN-based brain MRI study has been done to detect Alzheimer's disease. The suggested CNN model was not properly compared to pre-trained CNN models (InceptionV3, Xception, MobilenetV2, VGG). Hence, we provide a 12-layer CNN model for binary classification and Alzheimer's disease diagnosis using brain MRI data. Using the Open Access Series of Imaging Studies (OASIS) dataset, the proposed model is compared to existing CNN models in accuracy, precision, recall, F1 score, and ROC curve. The main contribution of the research is a 12-layer CNN model with an accuracy of 97.75%, which is greater than any other existing CNN models reported on this dataset. The publication also compares our model to pre-trained CNN models (InceptioV3, Xception, MobilenetV2, VGG). The proposed model outperforms existing models in experiments.

2.5 Multimodal Neuroimaging based Alzheimer's Disease Diagnosis using Evolutionary RVFL Classifier

Author Name : Tripti Goel, Rahul Sharma

Year of Publish : 2023

Consistent decline in mental faculties over time is a hallmark of Alzheimer's disease (AD), one of the most well-known forms of dementia. Mild cognitive impairment is a progressive disease that can only be reversed via early diagnosis and treatment (MCI). MRI and PET scans can detect the most frequent indicators used to diagnose Alzheimer's disease, including structural atrophy and the formation of plaques and tangles. Since early identification of this fatal neurodegenerative illness relies on incorporating structural and metabolic information, this paper suggests wavelet transform-based multimodality fusion of MRI and PET data. Additionally, the deep learning model, ResNet-50, retrieves the fused images' features. The collected features are then classified using a single-hidden-layer random-vector functional link (RVFL). An evolutionary method is being used to fine-tune the weights and biases of the original RVFL network. The effectiveness of the proposed algorithm is demonstrated by extensive testing and comparison using the ADNI dataset, which is available to the public.

2.6 Classification of Alzheimer's Disease from MRI Data Using an Ensemble of Hybrid Deep Convolutional Neural Networks

Author Name : Emimal Jabason, M. Omair Ahmad

Year of Publish : 2019

Although there is no cure for Alzheimer's disease (AD), an accurate early diagnosis is extremely important for both the patient and social care, and it will become even more significant once disease-modifying agents are available to prevent, cure, or even slow down the progression of the disease. In recent years, classification of AD through deep learning techniques has been one of the most active research areas in the medical field. However, most of the existing techniques cannot leverage the entire spatial information; hence, they lose the inter-slice correlation. In this paper, we propose a novel classification algorithm to discriminate patients having AD, mild cognitive impairment (MCI), and cognitively normal (CN) using an ensemble of hybrid deep learning architectures to leverage a more complete spatial information from the MRI data. The experimental results obtained by applying the proposed algorithm on the OASIS dataset show that the performance of the proposed classification framework to be superior to that of some conventional methods.

2.7 Transfer Learning for Alzheimer's Disease Detection on MRI Images

**Author Name : **Amir Ebrahimi-Ghahnavieh, Suhuai Luo

**Year of Publish : **2019

In this article, we use deep learning methods to MRI images in an effort to detect Alzheimer's disease. A significant difficulty in this area of study is the dearth of appropriate data for training a deep model. Based on our research, we know that subject classification using 2D convolutional neural networks and transfer learning is one of the current trends in Alzheimer's disease research. Each 3D MRI volume is then separated into 2D image slices such that a previously trained 2D convolutional neural network may be used to perform segmentation and classification on each slice individually. Nevertheless, the 2D convolutional neural network cannot take into account the interconnectedness of the 2D picture slices that make up an MRI volume. We suggest using a recurrent neural network after a convolutional neural network to learn the association between image sequences for each subject and to reach a conclusion using all input slices rather than just one. Our findings demonstrate that improving the accuracy of the entire system by training the recurrent neural network on features extracted by a convolutional neural network is possible.

2.8 Regression and Classification of Alzheimer’s Disease Diagnosis Using NMF-TDNet Features From 3D Brain MR Image

**Author Name : ** Huan Lao, Xuejun Zhang

Year of Publish : 2022

With deep learning and medical imaging technology, many researchers use convolutional neural network (CNN) to obtain deep-level medical image features to better classify Alzheimer's disease (AD) and predict clinical scores. The lightweight deep-learning network PCANet uses principal component analysis (PCA) to generate multilevel filter banks for centralised sample learning, binarizes, and generates blockwise histograms to obtain image features. PCANet's flexibility is reduced by the tens of thousands or hundreds of thousands of extracted PCANet features and the sample data-dependent formation of multilevel filter banks. This paper proposes the nonnegative matrix factorization tensor decomposition network, a data-independent network based on PCANet, to solve these issues (NMF-TDNet). Instead of PCA, we utilise nonnegative matrix factorization (NMF) to generate multilevel filter banks for sample learning, then use the learning results to build a higher-order tensor and perform tensor decomposition (TD) to reduce data dimensionality and provide picture features. Finally, our method uses these features as SVM input for AD classification diagnosis and clinical score prediction. Our method is extensively tested on ADNI-1, ADNI-2, and OASIS datasets. NMF-TDNet features as input outperformed PCANet features in data dimensionality reduction.

2.9 Deep Learning Framework for Alzheimer’s Disease Diagnosis via 3D-CNN and FSBi-LSTM

**Author Name : ** Chiyu Feng, Ahmed Elazab

Year of Publish : 2019

Neurodegenerative Alzheimer's disease (AD) is irreversible. Mild cognitive impairment (MCI) is the prodromal stage of AD, which can be progressive (pMCI) or stable (i.e., sMCI). Deep learning has helped convolutional neural networks (CNNs) recognise MRI and PET images for AD diagnosis. CNNs are difficult to employ for AD diagnosis due to the lack of imaging data. We create a new deep learning framework for this. Our approach takes advantage of 3D-CNN and FSSBi-LSTM. Initially, we create a 3D-CNN architecture for MRI and PET deep feature representation. FSBi-LSTM is then applied to deep feature maps' hidden spatial information to increase performance. The ADNI dataset validates our method. Our method beats comparable algorithms in differentiating AD, pMCI, and sMCI from NC with average accuracies of 94.82%, 86.36%, and 65.35%, respectively.

2.10 A CNN Model: Earlier Diagnosis and Classification of Alzheimer Disease using MRI

**Author Name : **Ahmad Waleed Salehi, Preety Baglat

Year of Publish : 2020

The most common form of dementia, Alzheimer's Disease (AD), damages brain cells and impairs memory and daily function. Utilizing MRI (Magnetic Resonance Imaging) scan brain images, we may use AI technology to diagnose, predict, and classify AD patients as having or not having this lethal disease. This is done to create the finest prediction and detection tools for radiologists, doctors, and carers to save time, money, and treat patients with this disease. DL algorithms operate well with huge datasets, making them beneficial for AD diagnosis in recent years. We used 1512 mild, 2633 normal, and 2480 AD MRI images from the ADNI 3 class to implement Convolutional Neural Network (CNN) for early diagnosis and classification of AD. The model performed well compared to another relevant research with 99% accuracy. We also compared the result with our previous work on OASIS dataset using machine learning algorithms, which showed that deep learning approaches may be better for large amounts of medical data.

CHAPTER 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Some of the symptoms of dementia include memory loss, trouble processing language, difficulties communicating, and changes in mood. The principal impacts of the condition on a patient's mind and body go hand in hand with a significant adjustment in lifestyle and the ability to carry out ordinary duties. The voxel, region, and patch-based techniques relied significantly on manually constructed features and feature representations since they saw segmentation jobs as classification problems. It took more time and a larger number of expertly segmented images to train classification algorithms.

3.2 DISADVANTAGES OF EXISTING SYSTEM

- minimal foresight.
- Existing method takes longer time to classify.

3.3 PROPOSED SYSTEM

Preprocessing of MR images, skull stripping (the extraction of essential parts of the brain by removing the brain membrane and unnecessary tissues), feature extraction and selection of non-redundant features from the processed MR images, application of fractal analysis on the selected features, and classification using the proposed machine learning method are all required for successful detection and classification of Alzheimer's disease from brain MR images.

3.4 FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

3.5 HARWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

- System: Intel core I3 processer 64 bits.
- Monitor: LED.
- Mouse: Logitech.
- Ram: 4.00 GB.

3.6 SOFTWARE REQUIREMENTS

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

- Operating system: Windows 64 bit
- Language: Python
- Platform: Anaconda3

3.7 TECHNOLOGIES USED

- Python
- Deep learning

3.8 PYTHON

Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.
Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open-source reference implementation.
A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

3.9 ANACONDA

Anaconda is a free open-source data science tool that focusses on the distribution of R and Python programming languages for data science and machine learning tasks. Anaconda aims at simplifying the data management and deployment of the same.

Anaconda is a powerful data science platform for data scientists. The package manager of Anaconda is the conda which manages the package versions.

Anaconda is a tool that offers all the required package involved in data science at once. The programmers choose Anaconda for its ease of use.

Anaconda is written in Python, and the worthy information on Conda is unlike pip in Python, this package manager checks for the requirement of the dependencies and installs it if it is required. More importantly, warning signs are given if the dependencies already exist.

Conda very quickly installs the dependencies along with frequent updates. It facilitates creation and loading with equal speed along with easy environment switching.

The installation of Anaconda is very easy and most preferred by non-programmers who are data scientists.

Anaconda is pre-built with more than 1500 Python or R data science packages. Anaconda has specific tools to collect data using Machine learning and Artificial Intelligence.

Anaconda is indeed a tool used for developing, testing and training in one single system. The tool can be managed with any project as the environment is easily manageable.

Anaconda is great for deep models and neural networks. You can build models, deploy them, and integrate with leading technologies in the subject. Anaconda is optimized to run efficiently for machine learning tasks and will save you time when developing great algorithms. Over 250 packages are included in the distribution. You can install other third-party packages through the Anaconda terminal with conda install. With over 7500 data science and machine learning packages available in their cloud-based repository, almost any package you need will be easily accessible. Anaconda offers individual, team, and enterprise editions. Included also is support for the R programming language.

The Anaconda distribution comes with packages that can be used on Windows, Linux, and MacOS. The individual edition includes popular package names like numpy, pandas, scipy, sklearn, tensorflow, pytorch, matplotlib, and more. The Anaconda Prompt and PowerShell make working within the filesystem easy and manageable. Also, the GUI interface on Anaconda Navigator makes working with everything exceptionally smooth. Anaconda is an excellent choice if you are looking for a thriving community of Data Scientists and ever-growing support in the industry. Conducting Data Science projects is an increasingly simpler task with the help of great tools like this.

CHAPTER 4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM). Superficially, DFDs can resemble flow charts or Unified Modeling Language (UML), but they are not meant to represent details of software logic. DFDs make it easy to depict the business requirements of applications by representing the sequence of process steps and flow of information using a graphical representation or visual representation rather than a textual description. When used through an entire development process, they first document the results of business analysis. Then, they refine the representation to show how information moves through, and is changed by, application flows. Both automated and manual processes are represented.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.OMG is continuously making efforts to create a truly industry standard. UML stands for Unified Modeling Language. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

4.2.1 USE CASE DIAGRAM

A use case diagram is a way to summarize details of a system and the users within that system. It is generally shown as a graphic depiction of interactions among different elements in a system. Use case diagrams will specify the events in a system and how those events flow, however, use case diagram does not describe how those events are implemented. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems. There are a number of benefits with having a use case diagram over similar diagrams such as flowcharts. It includes the following elements:

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.

- The use cases, which are the specific roles played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

4.2.2 SEQUENCE DIAGRAM

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when

CHAPTER 5
SYSTEM ARCHITECTURE

5. SYSTEM ARCHITECTURE

5.1 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components. An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element. Software environments are complex, and they aren't static.

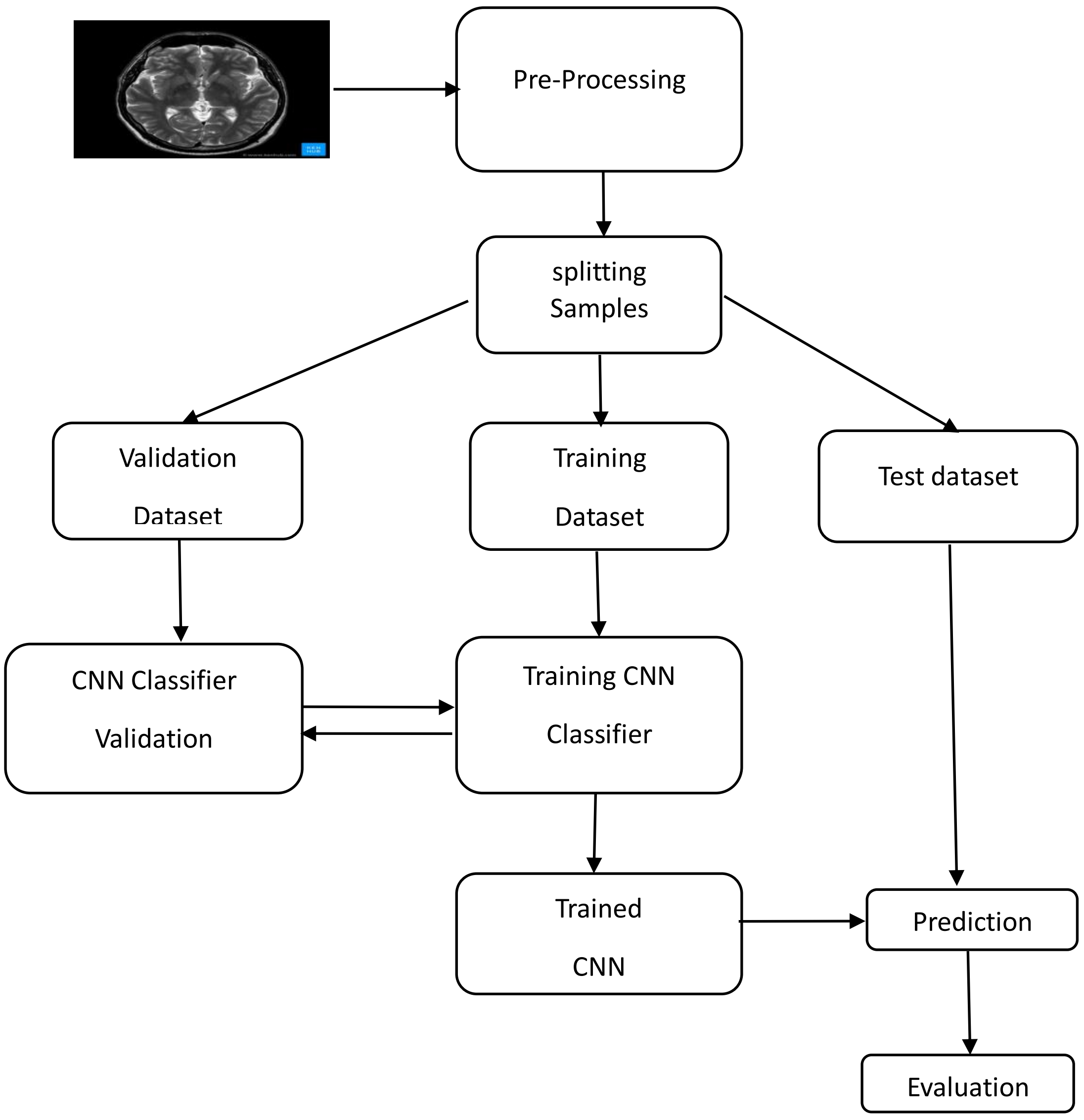


Fig 5.1 Architecture Diagram

5.2 ALGORITHM

5.2.1 CONVOLUTIONAL NEURAL NETWORK

A convolutional neural network (CNN or ConvNet) is a network architecture for deep learning that learns directly from data.

CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories. They can also be quite effective for classifying audio, time-series, and signal data.

A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.

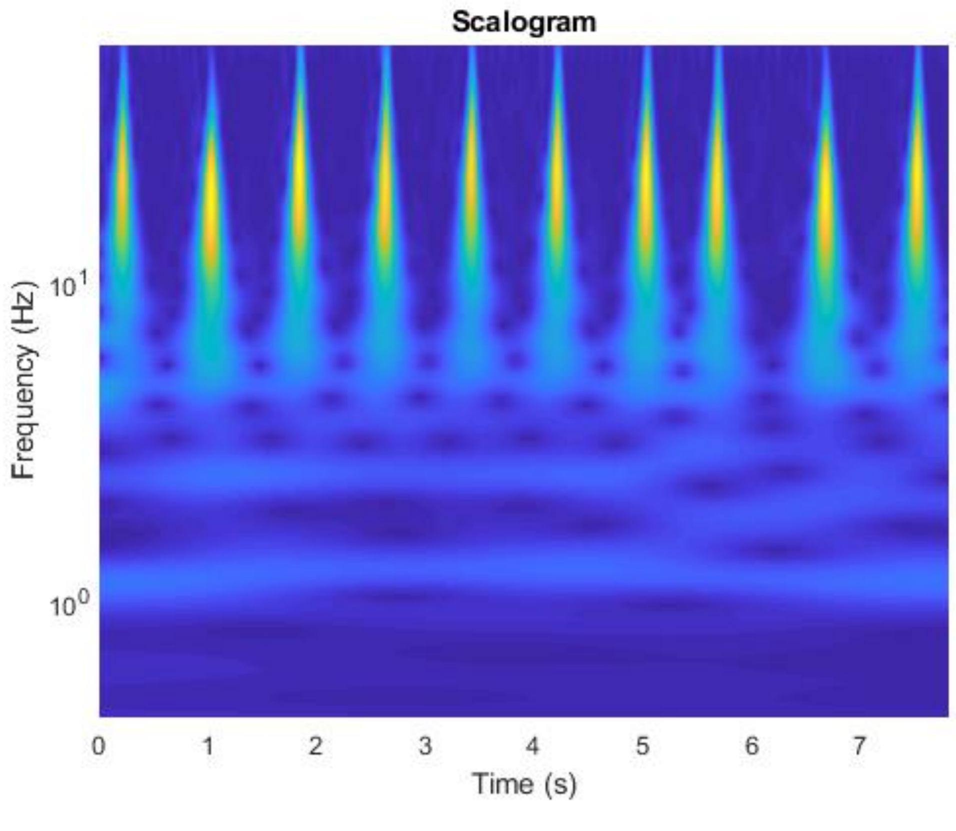


Fig 5.2.1.1 Classify Time Series Using Wavelet Analysis and Deep Learning

Consider using CNNs when you have a large amount of complex data (such as image data). You can also use CNNs with signal or time-series data when preprocessed to work with the network structure.

CHAPTER 6

SYSTEM IMPLEMENTATION

6. SYSTEM IMPLEMENTATION

6.1 MODULE DESIGN SPECIFICATION

The model for real-time image detection using CNN in Python involves several key steps:

- **Data collection and acquisition**
- **Data pre-processing**
- **Feature extraction**
- **Classification model**
- **Model evaluation**
- **Model deployment**

6.1.1 Data collection and acquisition:

The dataset used for training the CNN model should be representative of the image that need to be detected. The dataset can be collected from various sources, such as public datasets or custom data collection efforts. Data used in the preparation of this report were obtained from the publicly available Alzheimer's Disease Neuroimaging Initiative (ADNI) database and the Open Access Series of Imaging Studies (OASIS) project database. The most recent visit in which a diagnosis was made was considered the best available "ground-truth" to train the classifiers.

6.1.2 Data pre-processing:

This step involves pre-processing of the MRI images such as skull stripping, intensity normalization, and image registration. It is important to ensure that the images are of the same size, orientation, and spatial resolution to facilitate easy comparison and analysis.

6.1.3 Feature extraction:

This step involves extracting features from the pre-processed images, such as gray matter volume, cortical thickness, and hippocampal shape. These features are then used to classify the images into Alzheimer's disease (AD) like Mild Demented, Moderate Demented, Non Demented and Very Mild Demented.

6.1.4 Classification model:

The extracted features are then used to train a classification model, as convolutional neural network (CNN), to classify the images as Mild Demented, Moderate Demented, Non Demented and Very Mild Demented.

6.1.5 Model evaluation:

The trained model is evaluated using various metrics such as accuracy, sensitivity, specificity, and F1 score. The evaluation process helps to determine the effectiveness of the classification model and identify areas for improvement.

6.1.6 Model deployment:

The final step involves deploying the classification model to classify new MRI images into Mild Demented, Moderate Demented, Non Demented and Very Mild Demented. The model can be deployed as a standalone application, integrated into existing healthcare systems, or used as a diagnostic tool by healthcare professionals.

CHAPTER 7

SYSTEM TESTING

7. SYSTEM TESTING

7.1 SOFTWARE TESTING

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists.

7.1.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.1.2 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

- Valid Input: identified classes of valid input must be accepted.
- Invalid Input: identified classes of invalid input must be rejected.
- Functions: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.1.3 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

7.1.4 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.1.5 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works

7.1.6 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points

7.1.7 OUTPUT TESTING

• Output of test cases compared with the expected results created during design of test cases.

• Asking the user about the format required by them tests the output generated or displayed by the system under consideration.

• Here, the output format is considered into two was, one is on screen and another one is printed format.

• The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.

• The output comes out as the specified requirements as the user's hard copy.

7.1.8 USER ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

• Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.

• Two set of acceptance test to be run: 1. Those developed by quality assurance group 2. Those developed by customer

7.2 TEST CASES

TEST REPORT: 01

PRODUCT: AUTOMATED DETECTION OF ALZHEIMER DISEASE

USE CASE: UPLOAD IMAGE

TEST CASE ID	TEST CASE / ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS / FAIL
1	Upload the image as an input	Uploaded	As expected	PASS

Table-7.1 TEST CASE FOR IMAGE UPLOAD

TEST REPORT: 02

PRODUCT: AUTOMATED DETECTION OF ALZHEIMER DISEASE

USE CASE: IDENTIFY AND DETECT

TEST CASE ID	TEST CASE / ACTION TO BE PERFORMED	EXPECTED RESULT	ACTUAL RESULT	PASS / FAIL
1	Identify the intensity from the MRI image	Identified Successfully	As expected	PASS
2	Show the detected intensity from the image	Detected Successfully	As expected	PASS

Table-7.2 TEST CASE FOR IDETIFY AND DETECT

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

8. CONCLUSION AND FUTURE ENHANCEMENTS

8.1 CONCLUSION

As a result, we put our theory of transfer learning from CNN to other classifiers through its paces, following the steps in sequential order. It is possible for us to draw one of the defining qualities for training the classifier is the ability to transfer learnt parameters and features from a CNN that has been trained. If the feature transformation, selection, and classification processes are carried out in an intelligent manner, the CNN features trained classifier can achieve a higher level of performance than CNN networks itself. The performance of CNN can even be enhanced by properly modifying and tuning the architecture of CNN, as well as by appropriately optimizing the classification system.

8.2 FUTURE ENHANCEMENTS

APPENDICES

APPENDICES

A1. CODING

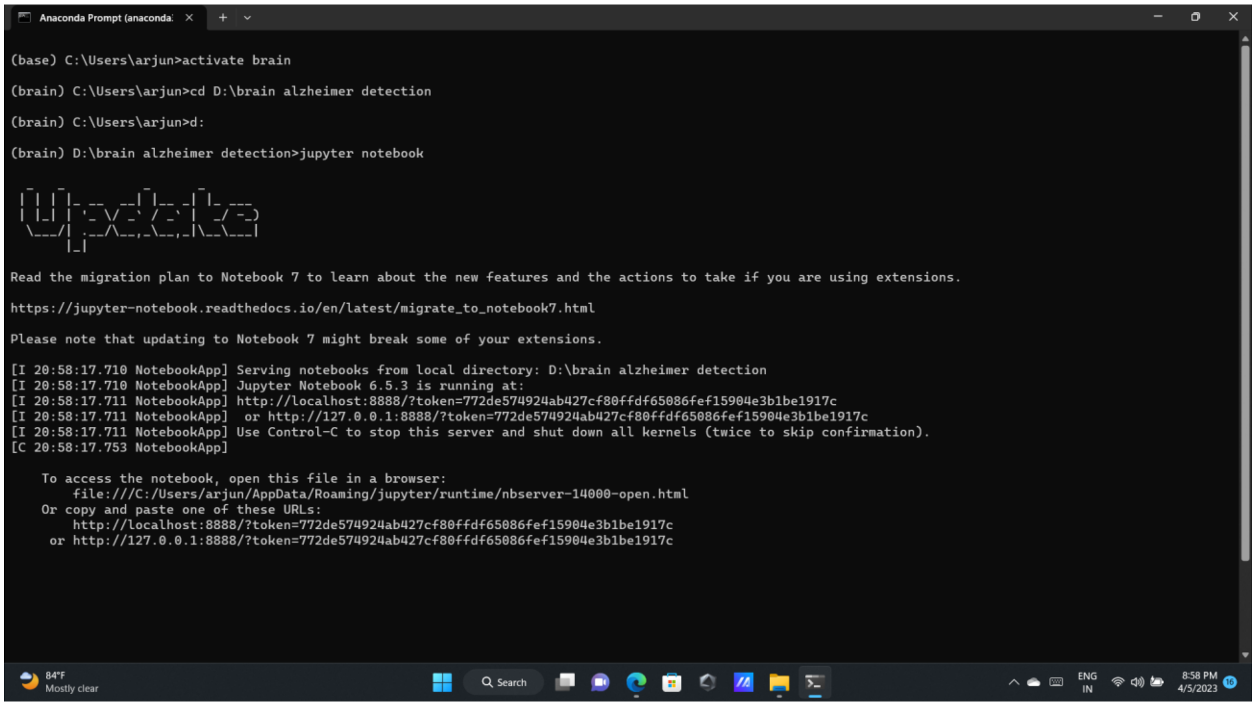
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import skimage.io
import os
import tqdm
import glob
import tensorflow

from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from skimage.color import rgb2gray

from skimage.io import imread, imshow
from skimage.transform import resize

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout, Flatten, Dense,
Activation, MaxPool2D, Conv2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications.densenet import DenseNet169
from tensorflow.keras.preprocessing.image import load_img, img_to_array

train_datagen = ImageDataGenerator(rescale = 1./255,
                                rotation_range=30,
                                zoom_range=0.2,
                                horizontal_flip=True,
                                vertical_flip=True,
                                validation_split = 0.2)

valid_datagen = ImageDataGenerator(rescale = 1./255,
                                validation_split = 0.2)

test_datagen  = ImageDataGenerator(rescale = 1./255)

train_dataset  = train_datagen.flow_from_directory(directory = 'D:/brain alzheimer
detection/Dataset',
                                               target_size = (224,224),
                                               class_mode = 'categorical',
                                               subset = 'training',
                                               batch_size = 128)
valid_dataset = valid_datagen.flow_from_directory(directory = 'D:/brain alzheimer
detection/Dataset',
                                              target_size = (224,224),
                                              class_mode = 'categorical',
                                              subset = 'validation',
                                              batch_size = 128)
fig, ax = plt.subplots(nrows = 1, ncols = 5, figsize=(20,20))

for i in tqdm(range(0,5)):
    rand1 = np.random.randint(len(train_dataset))
    rand2 = np.random.randint(100)
    ax[i].imshow(train_dataset[rand1][0][rand2])
    ax[i].axis('off')
    a = train_dataset[rand1][1][rand2]
    if a[0] == 1:
        ax[i].set_title('Mild Dementia')
    elif a[1] == 1:
        ax[i].set_title('Moderate Dementia')
    elif a[2] == 1:
        ax[i].set_title('Non Demetia')
    elif a[3] == 1:
        ax[i].set_title('Very Mild Dementia')

# Model Initialization

base_model = DenseNet169(input_shape=(224,224,3),
                         include_top=False,
                         weights="imagenet")

# Freezing Layers

for layer in base_model.layers:
    layer.trainable=False

# Building Model

model=Sequential()
model.add(base_model)
model.add(Dropout(0.5))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(2048,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4,activation='softmax'))

# Summary

model.summary()

# Model Compile

OPT    = tensorflow.keras.optimizers.Adam(lr=0.001)

model.compile(loss='categorical_crossentropy',
              metrics=[tensorflow.keras.metrics.AUC(name = 'auc')],
              optimizer=OPT)

# Defining Callbacks

filepath = './best_weights.hdf5'

earlystopping = EarlyStopping(monitor = 'val_auc',
                              mode = 'max' ,
                              patience = 15,
                              verbose = 1)

checkpoint    = ModelCheckpoint(filepath,
                               monitor = 'val_auc',
                               mode='max',
                               save_best_only=True,
                               verbose = 1)



callback_list = [earlystopping, checkpoint]

model_history=model.fit(train_dataset,
                        validation_data=valid_dataset,
                        epochs =30 ,
                        callbacks = callback_list,
                        verbose = 1)

# Summarize history for loss

plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
plt.show()
```

```
# Summarize history for loss

plt.plot(model_history.history['auc'])
plt.plot(model_history.history['val_auc'])
plt.title('Model AUC')
plt.ylabel('AUC')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
plt.show()

# Test Data

test_dataset  = test_datagen.flow_from_directory(directory = 'D:/brain alzheimer
detection/Dataset',
                                            target_size = (224,224),
                                            class_mode = 'categorical',
                                            batch_size = 128)

# Evaluating Loss and AUC

model.evaluate(test_dataset)

# Test Case 1: Non-Dementia

dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}
```

```
img = load_img('D:/brain alzheimer detection/Dataset/Moderate_Demented/moderate_9.jpg',
target_size = (224,224,3))

img = img_to_array(img)

img = img/255

imshow(img)

plt.axis('off')

img = np.expand_dims(img,axis=0)

answer = np.argmax(model.predict(img),axis=1)

probability = round(np.max(model.predict(img)*100),2)

print(probability, '% chances are there that the image is',idc[answer[0]])
```

A.2 SAMPLE SCREEN

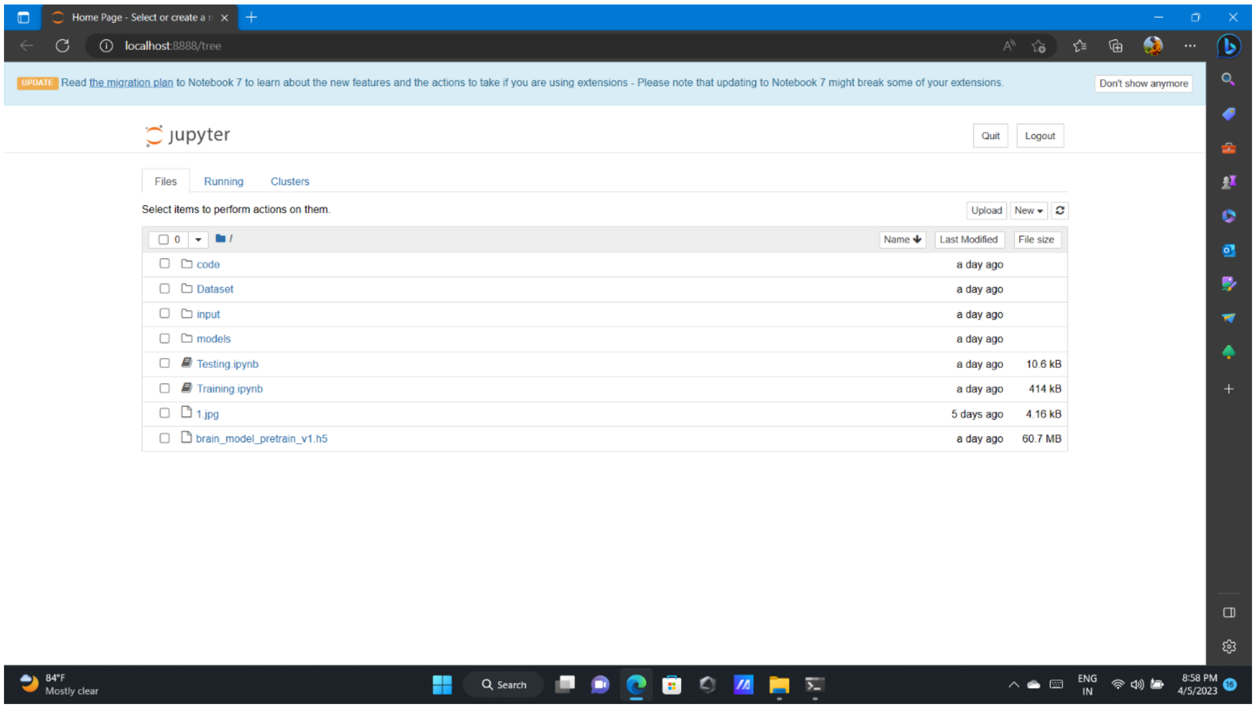
In fig A.2.1 Open Anaconda Prompt Console and activate brain. Change the directory to the location where the codes are stored. Call the Jupyter Notebook from here.



```
(base) C:\Users\arjun>activate brain

(brain) C:\Users\arjun>cd D:\brain alzheimer detection

(brain) C:\Users\arjun>d:

(brain) D:\brain alzheimer detection>jupyter notebook

  _   _          _      _
 | | | |_ __  __| |__ _| |_ ___
 | |_| | '_ \/ _` / _` |  _/ -_)
  \___/| .__/\__,_\__,_|\__\___|
       |_|

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions.

https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html

Please note that updating to Notebook 7 might break some of your extensions.

[I 20:58:17.710 NotebookApp] Serving notebooks from local directory: D:\brain alzheimer detection
[I 20:58:17.710 NotebookApp] Jupyter Notebook 6.5.3 is running at:
[I 20:58:17.711 NotebookApp] http://localhost:8888/?token=772de574924ab427cf80ffdf65086fef15904e3b1be1917c
[I 20:58:17.711 NotebookApp]  or http://127.0.0.1:8888/?token=772de574924ab427cf80ffdf65086fef15904e3b1be1917c
[I 20:58:17.711 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:58:17.753 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/arjun/AppData/Roaming/jupyter/runtime/nbserver-14000-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=772de574924ab427cf80ffdf65086fef15904e3b1be1917c
     or http://127.0.0.1:8888/?token=772de574924ab427cf80ffdf65086fef15904e3b1be1917c
```

Fig A.2.1

In fig A.2.2 This is the UI of the Jupyter Notebook , Here we can see all our project folders.

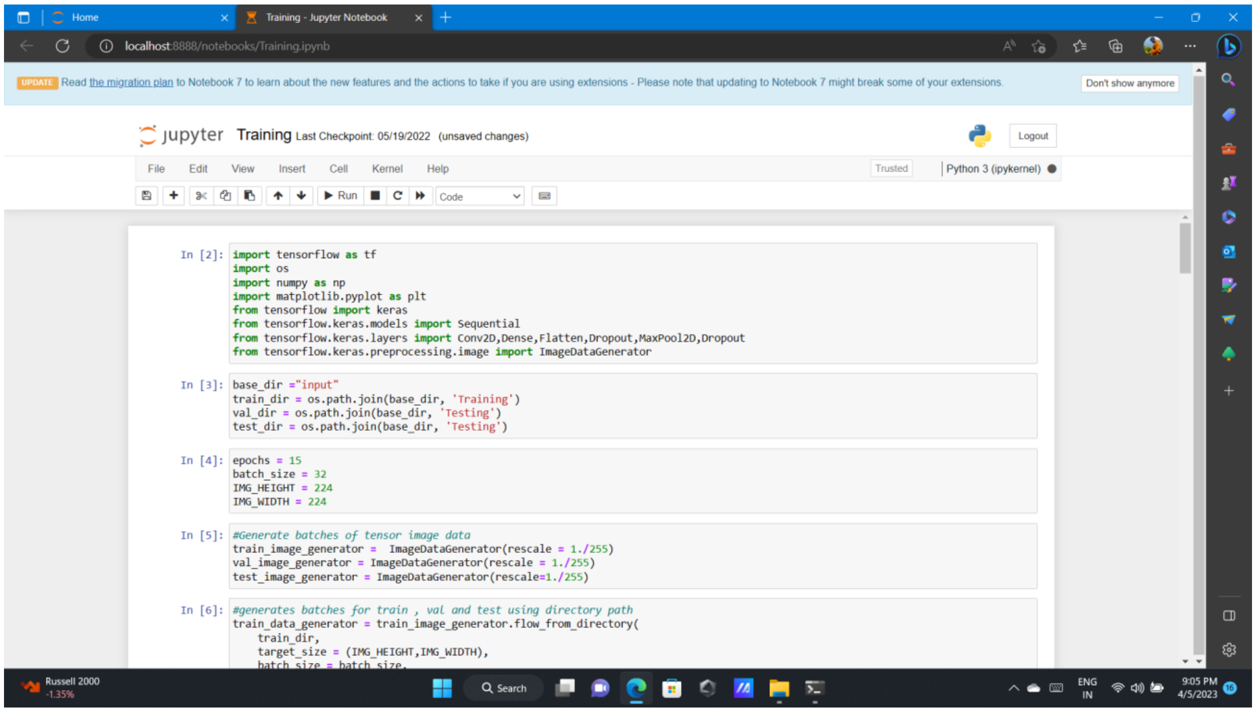


UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

Select items to perform actions on them.

Name	Last Modified	File size
code	a day ago	
Dataset	a day ago	
input	a day ago	
models	a day ago	
Testing.ipynb	a day ago	10.6 kB
Training.ipynb	a day ago	414 kB
1.jpg	5 days ago	4.16 kB
brain_model_pretrain_v1.h5	a day ago	60.7 MB

In fig A.2.3 Run all the cells in the Training folder (module)

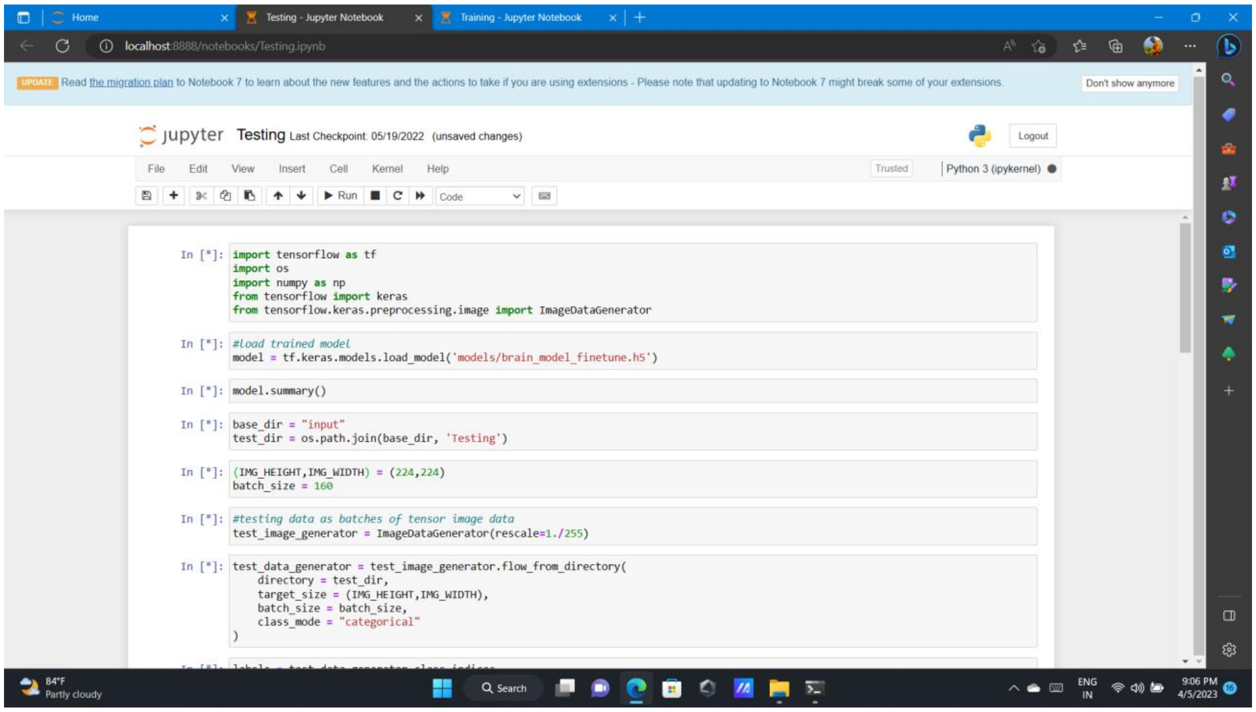

Training - Jupyter Notebook
localhost:8888/notebooks/Training.ipynb
jupyter Training Last Checkpoint: 05/19/2022 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3 (ipykernel)
Run
Code

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

```
In [2]: import tensorflow as tf
        import os
        import numpy as np
        import matplotlib.pyplot as plt
        from tensorflow import keras
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D,Dense,Flatten,Dropout,MaxPool2D,Dropout
        from tensorflow.keras.preprocessing.image import ImageDataGenerator

In [3]: base_dir ="input"
        train_dir = os.path.join(base_dir, 'Training')
        val_dir = os.path.join(base_dir, 'Testing')
        test_dir = os.path.join(base_dir, 'Testing')

In [4]: epochs = 15
        batch_size = 32
        IMG_HEIGHT = 224
        IMG_WIDTH = 224

In [5]: #Generate batches of tensor image data
        train_image_generator =  ImageDataGenerator(rescale = 1./255)
        val_image_generator = ImageDataGenerator(rescale = 1./255)
        test_image_generator = ImageDataGenerator(rescale=1./255)

In [6]: #generates batches for train , val and test using directory path
        train_data_generator = train_image_generator.flow_from_directory(
            train_dir,
            target_size = (IMG_HEIGHT,IMG_WIDTH),
            batch_size = batch_size,
```

Fig A.2.3

In fig A.2.4 Run all the cells in the Testing folder (module)

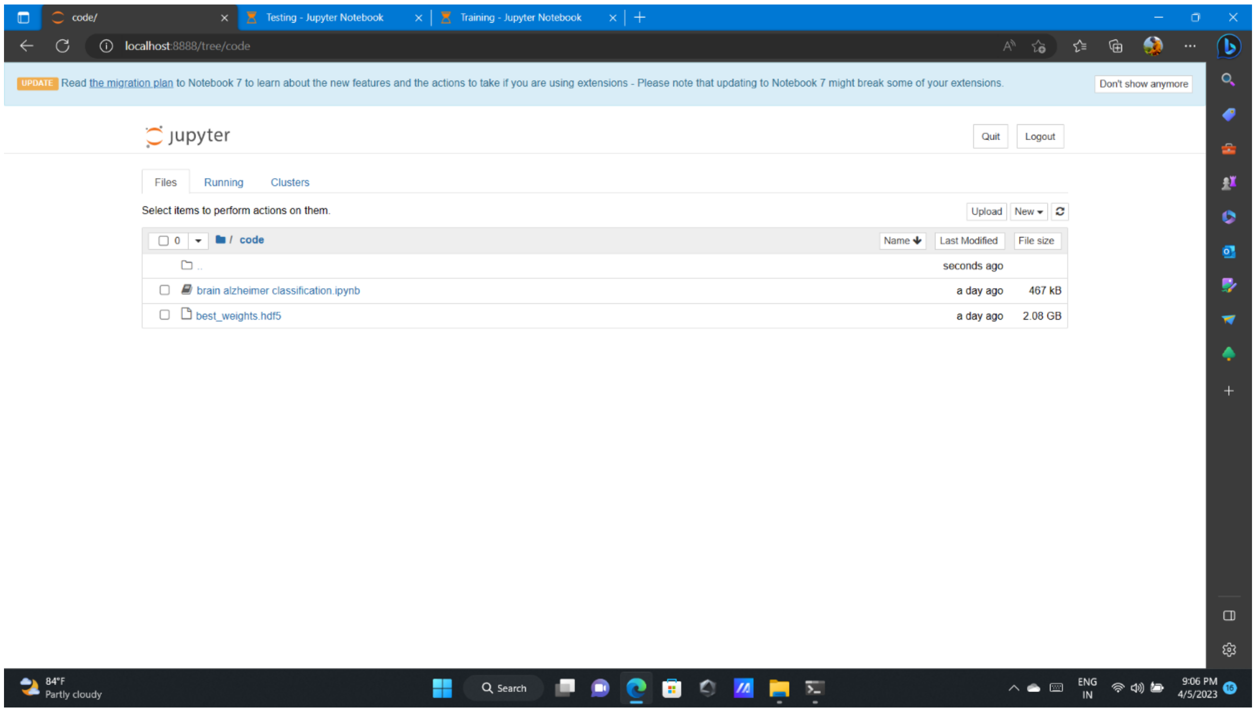

Testing - Jupyter Notebook
Training - Jupyter Notebook
localhost:8888/notebooks/Testing.ipynb
jupyter Testing Last Checkpoint: 05/19/2022 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3 (ipykernel)
Run
Code

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

```
In [*]: import tensorflow as tf
        import os
        import numpy as np
        from tensorflow import keras
        from tensorflow.keras.preprocessing.image import ImageDataGenerator

In [*]: #load trained model
        model = tf.keras.models.load_model('models/brain_model_finetune.h5')

In [*]: model.summary()

In [*]: base_dir = "input"
        test_dir = os.path.join(base_dir, 'Testing')

In [*]: (IMG_HEIGHT,IMG_WIDTH) = (224,224)
        batch_size = 160

In [*]: #testing data as batches of tensor image data
        test_image_generator = ImageDataGenerator(rescale=1./255)

In [*]: test_data_generator = test_image_generator.flow_from_directory(
            directory = test_dir,
            target_size = (IMG_HEIGHT,IMG_WIDTH),
            batch_size = batch_size,
            class_mode = "categorical"
        )
```

Fig A.2.4

In fig A.2.5 Now click on the code folder and select the main source file.



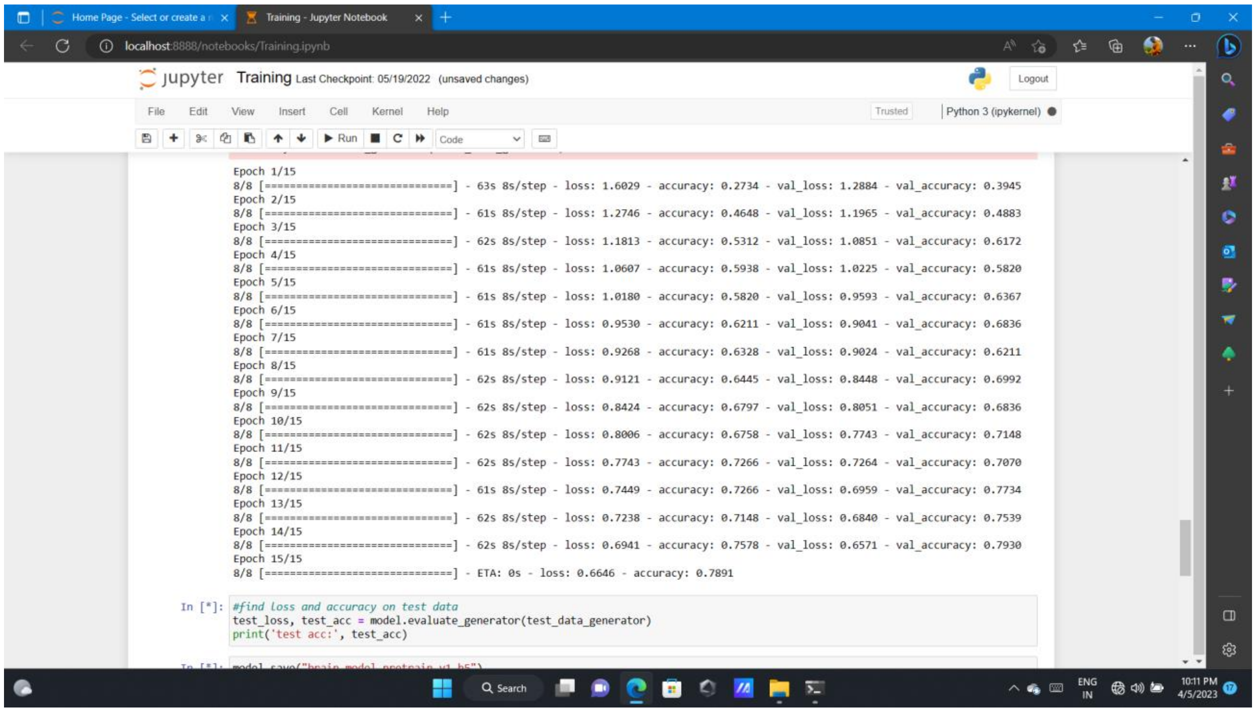
localhost:8888/tree/code

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Name	Last Modified	File size
/ code		
..	seconds ago	
brain alzheimer classification.ipynb	a day ago	467 kB
best_weights.hdf5	a day ago	2.08 GB

Fig A.2.5

In fig A.2.6 & A.2.7 This is how all the Epoches are completed after a successful run



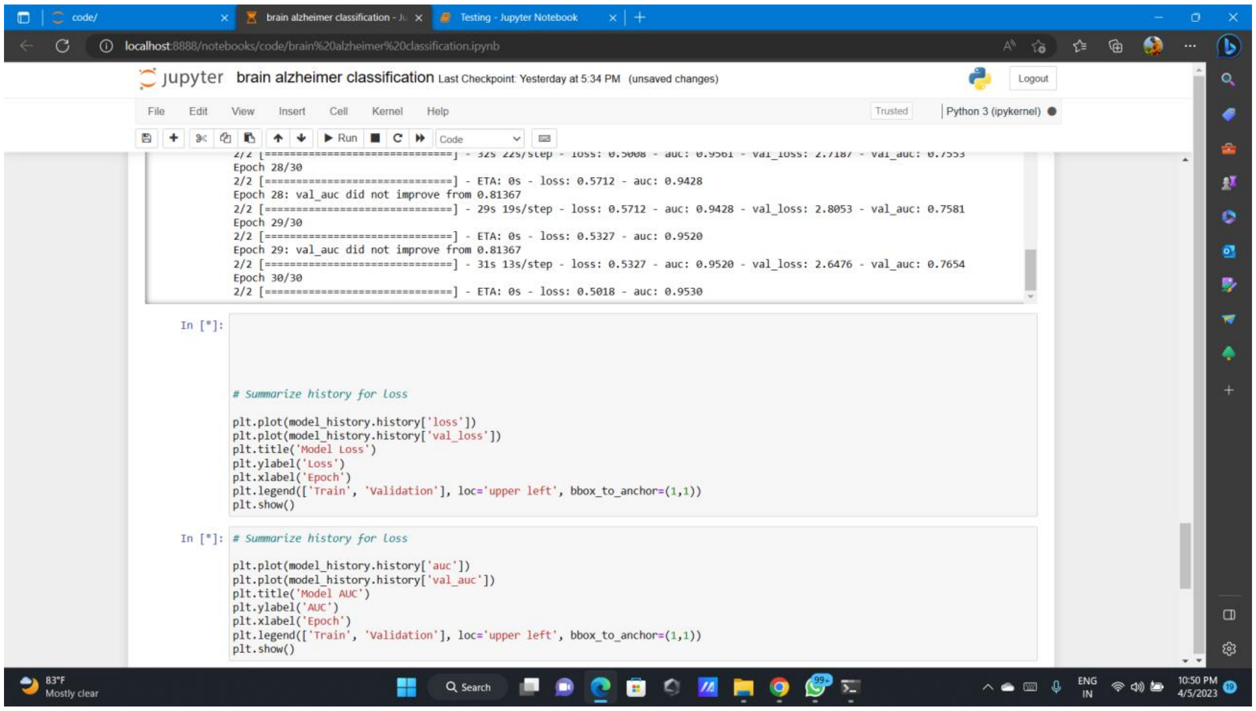
localhost:8888/notebooks/Training.ipynb

Training Last Checkpoint: 05/19/2022 (unsaved changes)

```
Epoch 1/15
8/8 [==============================] - 63s 8s/step - loss: 1.6029 - accuracy: 0.2734 - val_loss: 1.2884 - val_accuracy: 0.3945
Epoch 2/15
8/8 [==============================] - 61s 8s/step - loss: 1.2746 - accuracy: 0.4648 - val_loss: 1.1965 - val_accuracy: 0.4883
Epoch 3/15
8/8 [==============================] - 62s 8s/step - loss: 1.1813 - accuracy: 0.5312 - val_loss: 1.0851 - val_accuracy: 0.6172
Epoch 4/15
8/8 [==============================] - 61s 8s/step - loss: 1.0607 - accuracy: 0.5938 - val_loss: 1.0225 - val_accuracy: 0.5820
Epoch 5/15
8/8 [==============================] - 61s 8s/step - loss: 1.0180 - accuracy: 0.5820 - val_loss: 0.9593 - val_accuracy: 0.6367
Epoch 6/15
8/8 [==============================] - 61s 8s/step - loss: 0.9530 - accuracy: 0.6211 - val_loss: 0.9041 - val_accuracy: 0.6836
Epoch 7/15
8/8 [==============================] - 61s 8s/step - loss: 0.9268 - accuracy: 0.6328 - val_loss: 0.9024 - val_accuracy: 0.6211
Epoch 8/15
8/8 [==============================] - 62s 8s/step - loss: 0.9121 - accuracy: 0.6445 - val_loss: 0.8448 - val_accuracy: 0.6992
Epoch 9/15
8/8 [==============================] - 62s 8s/step - loss: 0.8424 - accuracy: 0.6797 - val_loss: 0.8051 - val_accuracy: 0.6836
Epoch 10/15
8/8 [==============================] - 62s 8s/step - loss: 0.8006 - accuracy: 0.6758 - val_loss: 0.7743 - val_accuracy: 0.7148
Epoch 11/15
8/8 [==============================] - 62s 8s/step - loss: 0.7743 - accuracy: 0.7266 - val_loss: 0.7264 - val_accuracy: 0.7070
Epoch 12/15
8/8 [==============================] - 61s 8s/step - loss: 0.7449 - accuracy: 0.7266 - val_loss: 0.6959 - val_accuracy: 0.7734
Epoch 13/15
8/8 [==============================] - 62s 8s/step - loss: 0.7238 - accuracy: 0.7148 - val_loss: 0.6840 - val_accuracy: 0.7539
Epoch 14/15
8/8 [==============================] - 62s 8s/step - loss: 0.6941 - accuracy: 0.7578 - val_loss: 0.6571 - val_accuracy: 0.7930
Epoch 15/15
8/8 [==============================] - ETA: 0s - loss: 0.6646 - accuracy: 0.7891
```

```
In [*]: #find loss and accuracy on test data
        test_loss, test_acc = model.evaluate_generator(test_data_generator)
        print('test acc:', test_acc)
```

Fig A.2.6



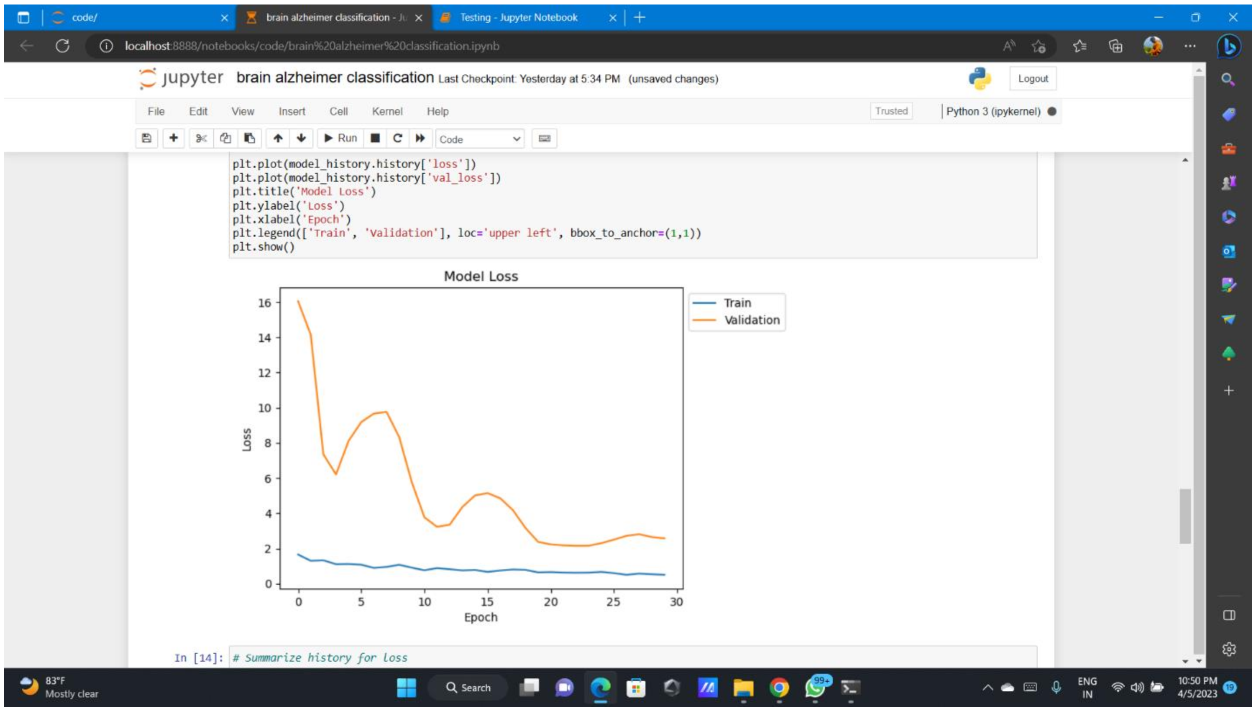
```
2/2 [==============================] - 32s 22s/step - loss: 0.5008 - auc: 0.9561 - val_loss: 2.7187 - val_auc: 0.7553
Epoch 28/30
2/2 [==============================] - ETA: 0s - loss: 0.5712 - auc: 0.9428
Epoch 28: val_auc did not improve from 0.81367
2/2 [==============================] - 29s 19s/step - loss: 0.5712 - auc: 0.9428 - val_loss: 2.8053 - val_auc: 0.7581
Epoch 29/30
2/2 [==============================] - ETA: 0s - loss: 0.5327 - auc: 0.9520
Epoch 29: val_auc did not improve from 0.81367
2/2 [==============================] - 31s 13s/step - loss: 0.5327 - auc: 0.9520 - val_loss: 2.6476 - val_auc: 0.7654
Epoch 30/30
2/2 [==============================] - ETA: 0s - loss: 0.5018 - auc: 0.9530
```

```
In [*]:



# Summarize history for loss

plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
plt.show()
```

```
In [*]: # Summarize history for loss

plt.plot(model_history.history['auc'])
plt.plot(model_history.history['val_auc'])
plt.title('Model AUC')
plt.ylabel('AUC')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
plt.show()
```

Fig A.2.7

In fig A.2.8 The Train and Validation graph is successfully plotted and shown below (Model Loss)

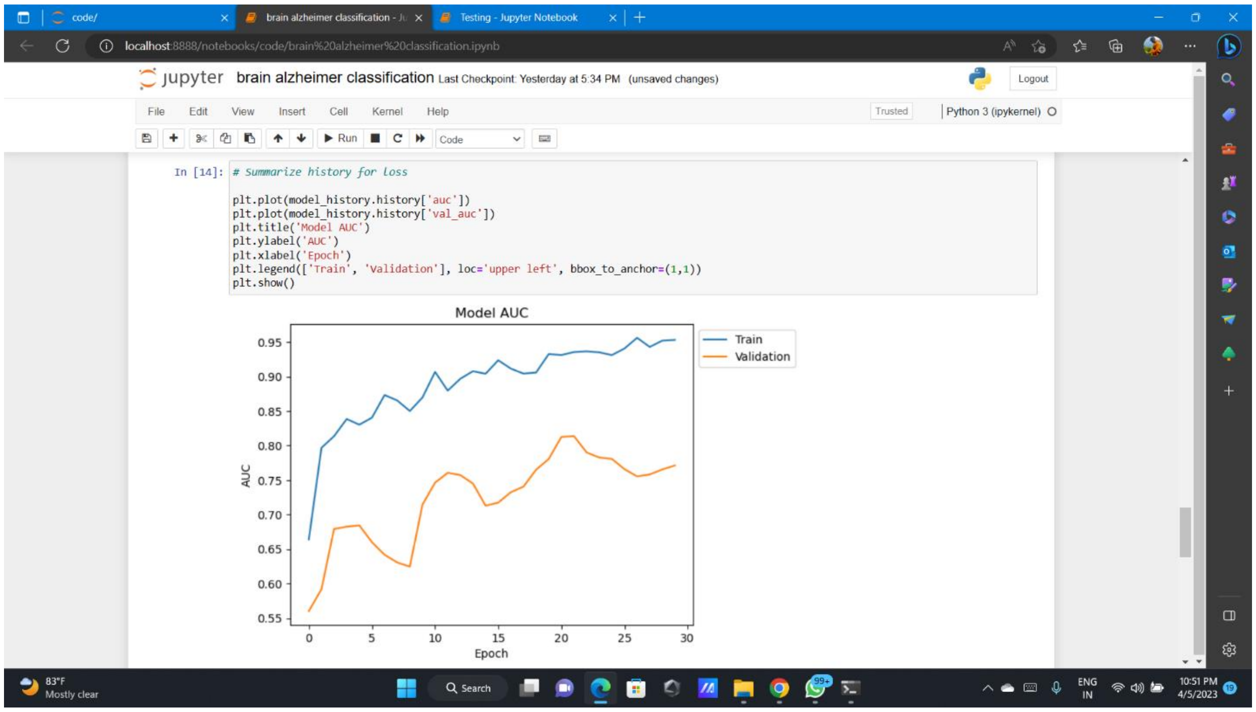


```
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
plt.show()
```

```
In [14]: # Summarize history for loss
```

Fig A.2.8

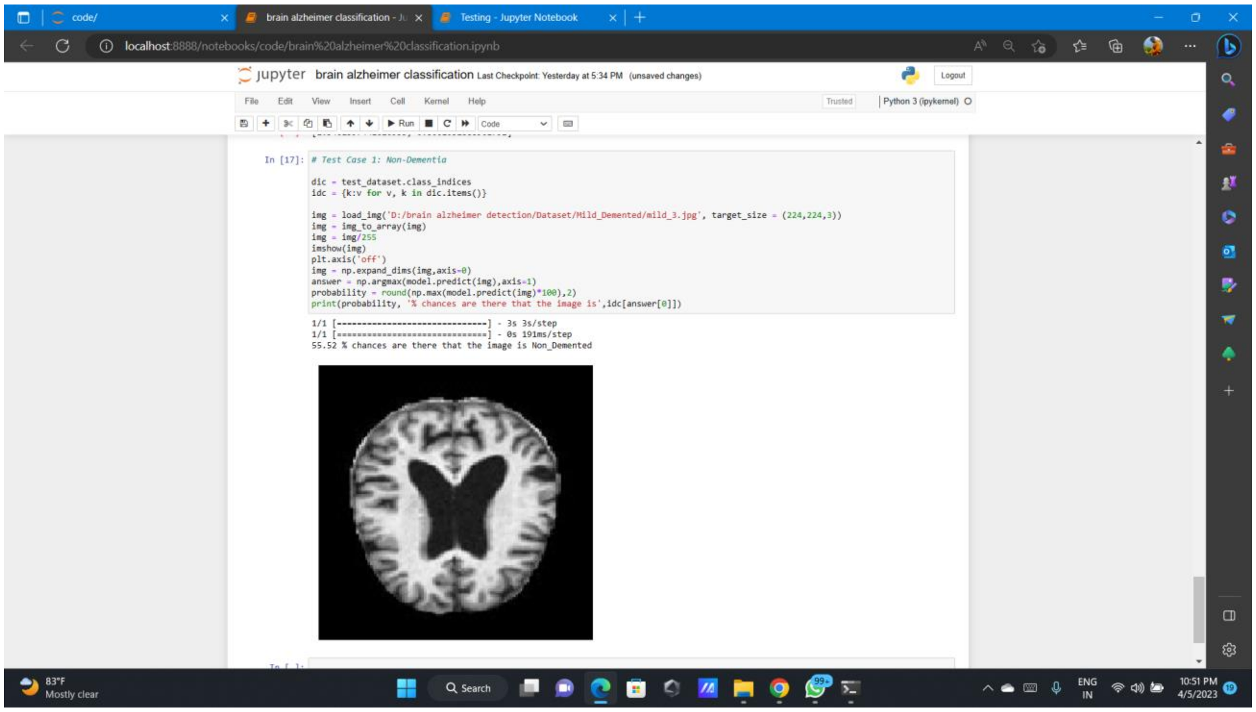
In fig A.2.9 The Train and Validation graph is successfully plotted and shown below (Model Accuracy)



```
In [14]: # Summarize history for Loss

         plt.plot(model_history.history['auc'])
         plt.plot(model_history.history['val_auc'])
         plt.title('Model AUC')
         plt.ylabel('AUC')
         plt.xlabel('Epoch')
         plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
         plt.show()
```

Fig A.2.9

In fig A.2.10 The final successful input and output are shown below



```
In [17]: # Test Case 1: Non-Dementia

         dic = test_dataset.class_indices
         idc = {k:v for v, k in dic.items()}

         img = load_img('D:/brain alzheimer detection/Dataset/Mild_Demented/mild_3.jpg', target_size = (224,224,3))
         img = img_to_array(img)
         img = img/255
         imshow(img)
         plt.axis('off')
         img = np.expand_dims(img,axis=0)
         answer = np.argmax(model.predict(img),axis=1)
         probability = round(np.max(model.predict(img)*100),2)
         print(probability, '% chances are there that the image is',idc[answer[0]])

         1/1 [==============================] - 3s 3s/step
         1/1 [==============================] - 0s 191ms/step
         55.52 % chances are there that the image is Non_Demented
```

Fig A.2.10

REFERENCES

REFERENCES

1. Y. Zhang, S. Wang, and Z. Dong, “Classification of Alzheimer disease based on structural magnetic resonance imaging by kernel support vector machine decision tree”, Progress In Electromagnetics Research, vol. 144, January 2014.
2. S. Chaplot, L. M. Patnaik, and N. R. Jagannathan, “Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network”, Biomedical Signal Processing and Control, vol. 1, pp. 86-92, January 2006.
3. M. Maitra, and A. Chatterjee, “A Slantlet transform based intelligent system for magnetic resonance brain image classification”, Biomedical Signal Processing and Control, vol. 1, pp. 299-306, October 2006.
4. E. S. A. El-Dahshan, T. Hosny, and A. B. M. Salem, “Hybrid intelligent techniques for MRI brain images classification”, Digital Signal Processing, vol. 1, pp. 299-306, October 2006.
5. Y. Zhang, L. Wu, and S. Wang, “Magnetic resonance brain image classification by an improved artificial bee colony algorithm”, Progress In Electromagnetics Research, vol. 116, pp. 65-79, 2011.
6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks”, Advances in neural information processing systems, pp. 1097- 1105, 2012.
7. K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition” arXiv:1512.03385, 2015.
8. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions”, In Proceedings of the IEEE conference on computer vision and pattern recognition pp. 1-9, 2015.
9. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587, 2014.
10. N. Tajbakhsh, J. Shin, S. Gurudu, C. Kendall, M. Gotway, and J. Liang, “Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?”, IEEE Transactions on Medical Imaging, vol. 35, pp. 1299-1312, May 2016.
11. H. Shin, H. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, and J. Yao, “Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset

Characteristics, and Transfer Learning", IEEE Transactions on Medical Imaging, vol. 35, pp. 1285-1298, May 2016.

12. H. R. Roth, L. Lu, J. Liu, J. Yao, A.Seff, K. Cherry, L. Kim, and Ronald M. Summers, "Improving Computer-Aided Detection Using Convolutional Neural Networks and Random View Aggregation", IEEE Transactions on Medical Imaging, vol. 35, pp. 1170-1181, May 2016.
13. A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition", Proc.IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, pp. 512-519, 2014.