

ALZHEIMER CLASSIFICATION USING TENSORFLOW AND KERAS MODEL

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Science degree in Computer Science

By
Jayashree.P (Reg. no. 39290037)

Nithisha.K (Reg.no.39290074)

Oviya.D (Reg. no. 39290076)



**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING**

SATHYABAMA

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

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

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

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **D. Oviya(Reg. no. 39290076)** **K.Nithisha(Reg. no. 39290074)** and **P.Jayashree(Reg. no. 39290037)** who carried out the project entitled “ **ALZHEIMER CLASSIFICATION USING TENSORFLOW AND KERAS MODEL**” under my supervision from **November 2021 to May 2022.**

Internal Guide

Dr. M. SELVI, M.E., Ph.D.,

Head of the Department

Dr. L. LAKSHAMANAN, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **D. Oviya (Reg. no. 39290076)** ,**K.Nithisha(Reg. no. 39290074)** and **P.Jayashree(Reg. no. 39290037)** hereby declare that the Project Report entitled **“ALZHEIMER CLASSIFICATION USING TENSORFLOW AND KERAS MODEL”** done by me under the guidance of **Dr. M. SELVI, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in **Computer Science**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing **Dr. L. Lakshmanan M.E., Ph.D.** , and **Dr.S.Vigneshwari M.E., Ph.D. Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. M. SELVI, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ALZHEIMER CLASSIFICATION USING TENSORFLOW AND KERAS MODEL

TABLE OF CONTENT

SL.NO	TITLE	PAGE.NO
01.	ABSTRACT	10
02.	EXISTING SYSTEM 2.1 DRAWBACKS	11
03.	INTRODUCTION 3.1 DATA SCIENCE 3.2 ARTIFICIAL INTELLIGENCE 3.2.1 NATURAL LANGUAGE PROCESSING (NLP)	12
04.	MACHINE LEARNING	16
05.	DEEP LEARNING 5.1 INTERPRETATIONS 5.2 DEEP LEARNING REVOLUTION	18
06.	PREPARING THE DATASET	23
07.	PROPOSED SYSTEM 7.1 ADVANTAGES	23
08.	LITERATURE SURVEY	24
09.	SYSTEM STUDY 9.1 PROJECT GOAL	28

	9.2 OBJECTIVES	
	9.3 SCOPE OF THE PROJECT	

10.	OUTLINE OF THE PROJECT	
	10.1 OVERVIEW OF THE SYSTEM	29
11.	PROJECT REQUIREMENT	
	11.1 FUNCTIONAL REQUIREMENT	30
	11.2 NON-FUNCTIONAL REQUIREMENT	
12.	FEASIBILITY STUDY	29
	12.1 DATAFLOW DIAGRAM	
13.	SYSTEM ARCHITECTURE	35
14.	WORKFLOW DIAGRAM	36
15.	USECASE DIAGRAM	37
16.	CLASS DIAGRAM	38
17.	ACTIVITY DIAGRAM	39
18.	SEQUENCE DIAGRAM	40
19.	E.R – DIAGRAM	41
20.	COLLABORATION DIAGRAM	42
21.	ENVIRONMENT REQUIREMENT	
	21.1 SOFTWARE REQUIREMENTS	43
	21.2 HARDWARE REQUIREMENTS	
22.	SOFTWARE DESCRIPTION	
	22.1 ANACONDA NAVIGATOR	43

	22.2 JUPYTER NOTEBOOK 22.3 PYCHARM	
23.	PYTHON 23.1 HISTORY 23.2 DESIGN PHILOSOPHY & FEATURE 23.3 SYNTAX AND SEMANTICS 23.4.INDENTATION 23.5 STATEMENTS AND CONTROL FLOW 23.6 EXPRESSIONS	54
24.	METHODOLOGY	64
25.	ARTIFICIAL NEURAL NETWORK (ANN)	70
26.	ARCHITECTURE OF CNN	72
27.	TYPES OF CNN 27.1 ALEX NET 27.2 LENET	78
28.	LIST OF MODULES	82
29.	MODULE DESCRIPTION 29.1 IMPORT THE GIVEN IMAGE FROM DATASET 29.2 TO TRAIN THE MODULE BY GIVEN IMAGE DATASET 29.3 WORKING PROCESS OF LAYERS IN CNN MODEL 29.4 CLASSIFICATION IDENTIFICATION	82
30.	DEPLOY	90

	30.1 DJANGO 30.2 HTML 30.3 CSS	90
31.	CODING	102
32.	CONCLUSION	125
33.	FUTURE WORK	125

LIST OF FIGURES

SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	35
02	WORKFLOW DIAGRAM	36
03	USECASE DIAGRAM	37
04	CLASS DIAGRAM	38
05	ACTIVITY DIAGRAM	39
06	SEQUENCE DIAGRAM	40
07	ER – DIAGRAM	41

CHAPTER 1

1. ABSTRACT:

The Alzheimer's disease has been linked to a deficiency in the brain neurotransmitter acetylcholine. It is a progressive disease beginning with mild memory loss and possibly leading to loss of the ability to carry on a conversation and respond to the environment. Alzheimer's disease is a brain disorder that slowly destroys memory and thinking skills and, eventually, the ability to carry out the simplest tasks. Alzheimer's disease involves parts of the brain that control thought, memory, and language. Alzheimer treatment options depend on the alzheimer or not. The classification of alzheimer is performed by biopsy, which is not usually conducted before definitive brain surgery. The improvement of technology and deep learning can help neurologists in alzheimer diagnostics without invasive measures. A deep-learning algorithm that has achieved substantial results in image classification is the convolutional neural network (CNN). It is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classify successfully alzheimer normal and abnormal image.

Keywords: Alzheimer, deep learning, TensorFlow, CNN

CHAPTER 2

2. EXISTING SYSTEM:

This article presents an improved method of measuring technology similarity by introducing a subject-action-object (SAO) analysis that uses the feature weights of semantic structure and professional vocabulary to measure technology similarity in the medical field. First, the SAO semantic structures are extracted and cleaned; then the structures related to technology are identified using a semantic network of the unified medical language system (UMLS). Second, the similarity between the SAO semantic structures is evaluated using semantic information from the Metathesaurus of the UMLS. Technology similarity analysis can be used to identify competitors and search for partners for commercial enterprises. Therefore, this article develops an improved method to measure technology similarity in the medical field. The proposed indicator, TFW as technology feature weights of SAO, represents the importance of technology information. The UMLS measures the similarity between the SAO semantic structures.

2.1 Drawback:

- It has not focused on identifying CNN as classifier.
- It has not focused on increasing the recognition rate and classification accuracy of severity of alzheimer.
- Deployment of the model was not implemented.

CHAPTER 3

3. INTRODUCTION

3.1 Data Science:

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

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

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

Data Scientist:

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

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

3.2 ARTIFICIAL INTELLIGENCE:

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

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

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human

speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go). As machines become increasingly capable, tasks considered to require “intelligence” are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an “AI winter”), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field’s long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence “can be so precisely described that a machine can be made to simulate it”. This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure

relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

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

3.2.1 Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. “Keyword spotting” strategies for search are popular and scalable but dumb; a search query for “dog” might only match documents with the literal word “dog” and miss a document with the word “poodle”. “Lexical affinity” strategies use the occurrence of words such as “accident” to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of “narrative” NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

4. MACHINE LEARNING:

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data

and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

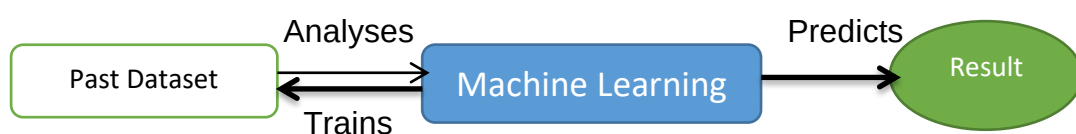


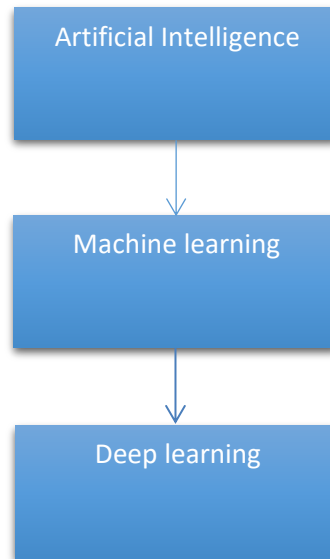
Fig: Process of Machine learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and

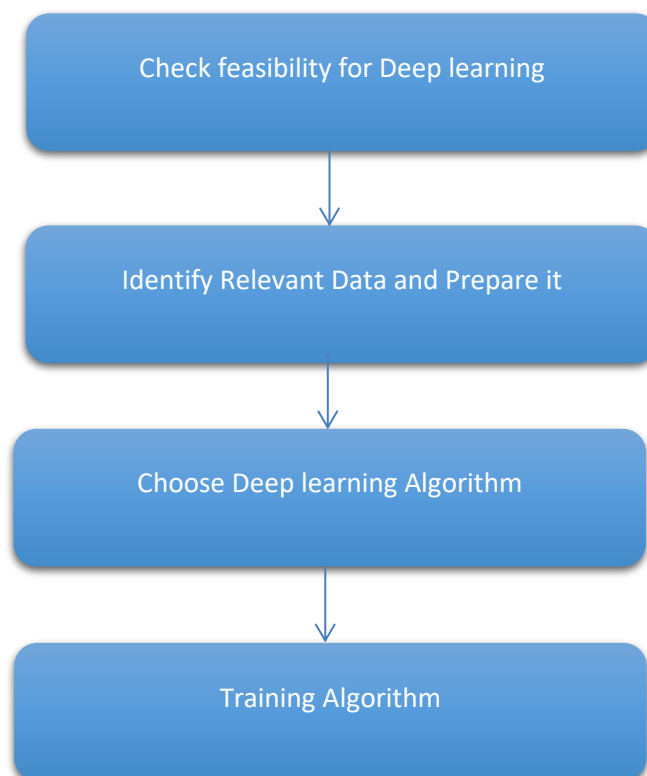
an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

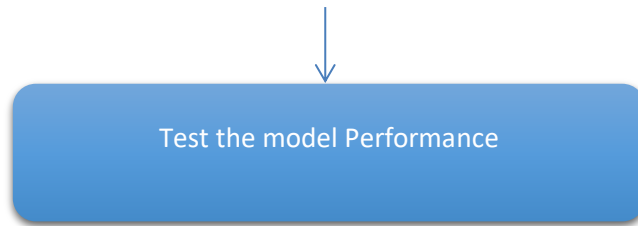
5. DEEP LEARNING

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. A formal definition of deep learning is- neurons Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how it recreates these neurons in a computer. So, it creates an artificial structure called an artificial neural net where we have nodes or neurons. It has some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.



It need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). It needs to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Choose the Deep Learning Algorithm appropriately. Algorithm should be used while training the dataset. Final testing should be done on the dataset.





Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analogue.

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, but that a network with a non-polynomial activation function with one hidden layer of unbounded width can. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically

informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

5.1 Interpretations:

Deep neural networks are generally interpreted in terms of the universal approximation theorem or probabilistic inference.

The classic universal approximation theorem concerns the capacity of feed-forward neural networks with a single hidden layer of finite size to approximate continuous functions. In 1989, the first proof was published by George Cybenko for sigmoid activation functions and was generalised to feed-forward multi-layer architectures in 1991 by Kurt Hornik. Recent work also showed that universal approximation also holds for non-bounded activation functions such as the rectified linear unit.

The universal approximation theorem for deep neural networks concerns the capacity of networks with bounded width but the depth is allowed to grow proved that if the width of a deep neural network with ReLU activation is strictly larger than the input dimension, then the network can approximate any Lebesgue integrable function; If the width is smaller or equal to the input dimension, then deep neural network is not a universal approximator.

The probabilistic interpretation derives from the field of machine learning. It features inference, as well as the optimization concepts of training and testing, related to fitting and generalization, respectively. More specifically, the probabilistic interpretation considers the activation nonlinearity as a cumulative distribution function. The probabilistic interpretation led to the introduction of dropout as regularizer in neural networks. The probabilistic interpretation was introduced by researchers including Hopfield, Widrow and Narendra and popularized in surveys such as the one by Bishop.

5.2 Deep learning revolution:

In 2012, a team led by George E. Dahl won the "Merck Molecular Activity Challenge" using multi-task deep neural networks to predict the biomolecular target of one drug. In 2014, Hochreiter's group used deep learning to detect off-target and toxic effects of environmental chemicals in nutrients, household products and drugs and won the "Tox21 Data Challenge" of NIH, FDA and NCATS.

Significant additional impacts in image or object recognition were felt from 2011 to 2012. Although CNNs trained by back-propagation had been around for decades, and GPU implementations of NNs for years, including CNNs, fast implementations of CNNs on GPUs were needed to progress on computer vision. In 2011, this approach achieved for the first time superhuman performance in a visual pattern recognition contest. Also in 2011, it won the ICDAR Chinese handwriting contest, and in May 2012, it won the ISBI image segmentation contest. Until 2011, CNNs did not play a major role at computer vision conferences, but in June 2012, a paper by Ciresan et al. at the leading conference CVPR showed how max-pooling CNNs on GPU can dramatically improve many vision benchmark records. In October 2012, a similar system by Krizhevsky et al. won the large-scale ImageNet competition by a significant margin over shallow machine learning methods. In November 2012, Ciresan et al.'s system also won the ICPR contest on analysis of large medical images for cancer detection, and in the following year also the MICCAI Grand Challenge on the same topic. In 2013 and 2014, the error rate on the ImageNet task using deep learning was further reduced, following a similar trend in large-scale speech recognition.

Image classification was then extended to the more challenging task of generating descriptions (captions) for images, often as a combination of CNNs and LSTMs.

Some researchers state that the October 2012 ImageNet victory anchored the start of a "deep learning revolution" that has transformed the AI industry.

In March 2019, Yoshua Bengio, Geoffrey Hinton and Yann LeCun were awarded the Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.

6. PREPARING THE DATASET:

This dataset contains approximately 400 train and 100 test images, which were then classified into 2 classes:

- DEMENTED
- NON DEMENTED

CHAPTER 4

7. PROPOSED SYSTEM:

- The classification of alzheimer is performed in a comparable way. The improvement of technology and deep learning can help neurologists in alzheimer diagnostics without invasive measures. A deep-learning algorithm that has achieved substantial results in image classification is the convolutional neural network (CNN).
- The classification was performed using a T1-weighted contrast-enhanced MRI image database which contains Alzheimer or not. As input, we used whole images, so it was not necessary to perform any preprocessing of the alzheimer. Samples of more number of images are collected that comprised of different classes such as normal and abnormal. Different number of images is collected for each class that was classified into input images.
- We proposed a Deep Learning (DL) based alzheimer prediction method to predict the result. The DL method used in the study is the Convolutional Neural Network (CNN). It is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classify successfully alzheimer.
- For deployment, we'll show the prediction result in local-host web application.

7.1 Advantages:

- To identify the Alzheimer disease easily and reduce the workload of doctors in the medical field.
- It is best model for deep learning technique to easily identify the alzheimer.

CHAPTER 5

8. LITERATURE SURVEY

General

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

Review of Literature Survey

Title : A Deep Learning Model to Predict a Diagnosis of Alzheimer Disease by Using 18F-FDG PET of the Brain

Author: Yiming Ding • Jae Ho Sohn, MD, MS • Michael G. Kawczynski, MS • Hari Trivedi, MD

Year : 2013

To develop and validate a deep learning algorithm that predicts the final diagnosis of Alzheimer disease (AD), mild cognitive impairment, or neither at fluorine 18 (18F) fluorodeoxyglucose (FDG) PET of the brain and compare its performance to that of radiologic readers. Prospective 18F-FDG PET brain images from the Alzheimer's Disease Neuroimaging Initiative (ADNI) (2109 imaging studies from 2005 to 2017, 1002 patients) and retrospective independent test set (40 imaging studies from 2006 to 2016, 40 patients) were collected. Final clinical diagnosis at follow-up was recorded. Convolutional neural network of InceptionV3 architecture was trained on 90% of ADNI data set and tested on the remaining 10%, as well as the independent test set, with performance compared to radiologic readers. Model was analyzed with sensitivity, specificity, receiver operating characteristic (ROC), saliency map, and t-distributed stochastic neighbor embedding.

Title : DeepAD: Alzheimer's Disease Classification via Deep Convolutional Neural Networks using MRI and fMRI

Author: Saman Sarraf, Danielle D.DeSouza

Year : 2016

To extract patterns from neuroimaging data, various statistical methods and machine learning algorithms have been explored for the diagnosis of Alzheimer's disease among older adults in both clinical and research applications; however, distinguishing between Alzheimer's and healthy brain data has been challenging in older adults (age > 75) due to highly similar patterns of brain atrophy and image intensities. Recently, cutting-edge deep learning technologies have rapidly expanded into numerous fields, including medical image analysis. This paper

outlines state-of-the-art deep learning-based pipelines employed to distinguish Alzheimer's magnetic resonance imaging (MRI) and functional MRI (fMRI) from normal healthy control data for a given age group. Using these pipelines, which were executed on a GPU-based high-performance computing platform, the data were strictly and carefully preprocessed. Next, scale- and shift-invariant low- to high-level features were obtained from a high volume of training images using convolutional neural network (CNN) architecture. In this study, fMRI data were used for the first time in deep learning applications for the purposes of medical image analysis and Alzheimer's disease prediction. These proposed and implemented pipelines, which demonstrate a significant improvement in classification output over other studies, resulted in high and reproducible accuracy rates of 99.9% and 98.84% for the fMRI and MRI pipelines, respectively. Additionally, for clinical purposes, subject-level classification was performed, resulting in an average accuracy rate of 94.32% and 97.88% for the fMRI and MRI pipelines, respectively. Finally, a decision making algorithm designed for the subject-level classification improved the rate to 97.77% for fMRI and 100% for MRI pipelines.

Title : Classification of Alzheimer's disease in MobileNet

Author: Xiaoling Lu, Haifeng Wu * and Yu Zeng

Year : 2018

As the aging of Chinese society becomes more and more serious, the number of elderly people has increased dramatically. At the same time, the number of patients with Alzheimer's disease (AD) has increased. At present, the main diagnostic method for Alzheimer's disease relies on experienced radiologists to analyse brain structural nuclear magnetic resonance (MRI) images to judge the condition, but this method is time-consuming and labor-intensive, and there is a certain subjectivity. This may cause misdiagnosis. By classifying the MRI images of patients with Alzheimer's disease and healthy controls (NC), for image classification, the Convolutional Neural Network (CNN) in deep learning has outstanding performance and accurate classification. The VGG 16 network model and the MobileNet network model of the convolutional neural network are compared, using

deep learning and transfer learning. We can find that the MobileNet network model is superior to the VGG 16 network model in classification accuracy.

Title : Classification of Alzheimer's Disease Using fMRI Data and Deep Learning Convolutional Neural Networks

Author: Saman Sarraf, Ghassem Tofighi

Year : 2016

Over the past decade, machine learning techniques especially predictive modeling and pattern recognition in biomedical sciences from drug delivery system [7] to medical imaging has become one of the important methods which are assisting researchers to have deeper understanding of entire issue and to solve complex medical problems. Deep learning is power learning machine learning algorithm in classification while extracting high-level features. In this paper, we used convolutional neural network to classify Alzheimer's brain from normal healthy brain. The importance of classifying this kind of medical data is to potentially develop a predict model or system in order to recognize the type disease from normal subjects or to estimate the stage of the disease. Classification of clinical data such as Alzheimer's disease has been always challenging and most problematic part has been always selecting the most discriminative features. Using Convolutional Neural Network (CNN) and the famous architecture LeNet-5, we successfully classified functional MRI data of Alzheimer's subjects from normal controls where the accuracy of test data on trained data reached 96.85%. This experiment suggests us the shift and scale invariant features extracted by CNN followed by deep learning classification is most powerful method to distinguish clinical data from healthy data in fMRI. This approach also enables us to expand our methodology to predict more complicated systems.

Title : A Novel Deep Learning Based Multi-class Classification Method for Alzheimer's Disease Detection Using Brain MRI Data

Author: Jyothi Islam, Yanqing ZHnag

Year : 2017

Alzheimer's Disease is a severe neurological brain disorder. It destroys brain cells causing people to lose their memory, mental functions and ability to continue daily activities. Alzheimer's Disease is not curable, but earlier detection can help improve symptoms in a great deal. Machine learning techniques can vastly improve the process for accurate diagnosis of Alzheimer's Disease. In recent days deep learning techniques have achieved major success in medical image analysis. But relatively little investigation has been done to applying deep learning techniques for Alzheimer's Disease detection and classification. This paper presents a novel deep learning model for multi-Class Alzheimer's Disease detection and classification using Brain MRI Data. We design a very deep convolutional network and demonstrate the performance on the Open Access Series of Imaging Studies (OASIS) database.

CHAPTER 6

9. SYSTEM STUDY

9.1 Project Goals:

- Load the data
 - Loading the given dataset
 - Import required libraries packages
- Pre-process the data
 - Reshape, data augmentations
- Define model
 - Sequential or Functional
 - Number of layers to be used, Number of nodes to be used in the model, Evaluation metrics

- Compile the model
 - Define loss function, optimizer, weights and bias
- Fit the model
 - Train data, Test data, epoch, Batchsize.

9.2 Objectives:

The goal is to develop a deep learning model for Alzheimer disease image classification by convolutional neural network algorithm for potentially classifying the results in the form of best accuracy by comparing the CNN architectures.

9.3 Scope:

The scope of this paper is to implement and investigate how different CNN methods impact the prediction. The scope in our research is to split this classes into Alzheimer defected and normal images. This can be classified into 2 classes. Besides, extracting hand crafted features from raw images after different processing is one of the best scopes we have worked in this research. The objective of using this method is to get some features which are specifically responsible for the Alzheimer disease prediction.

10.OUTLINE OF THE PROJECT

10.1 Overview of the system:

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

The steps involved in Building the data model is depicted below.

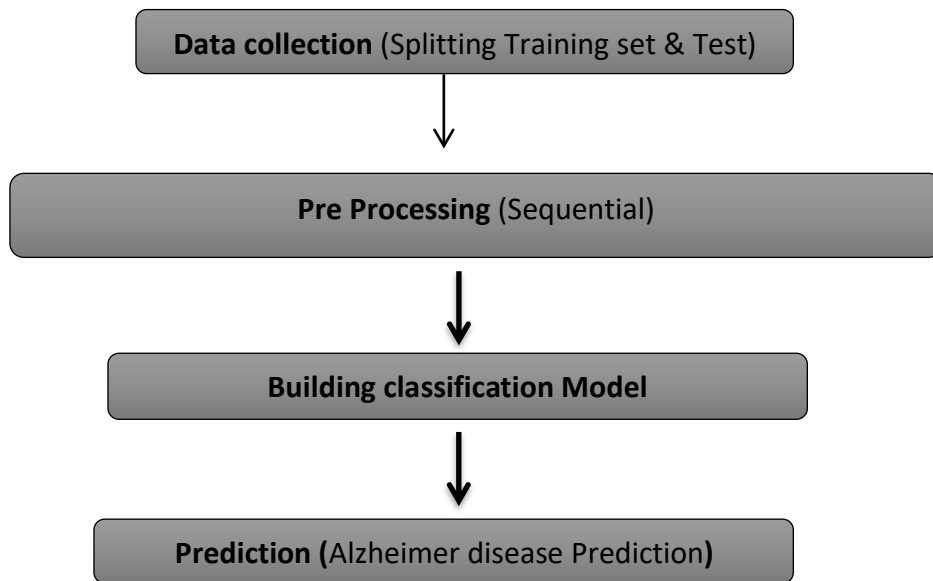


Fig: data flow diagram for CNN model

11.PROJECT REQUIREMENTS

General:

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
 - A. Hardware requirements
 - B. software requirements

11.1 Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis

process. It lists requirements of a particular software system. The following details to follow the special libraries like tensorflow, keras, matplotlib.

11.2 Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithm
4. Improving results
5. Prediction the result

12. FEASIBILITY STUDY

Splitting the dataset:

The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensor flow library in Python using the Keras method.

Construction of a Detecting Model:

Deep learning needs data gathering have lot of past image data's. Training and testing this model working and predicting correctly.

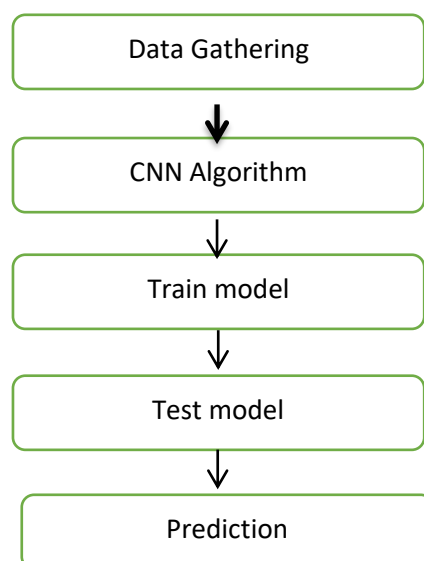


Fig: Steps of dataflow diagram

12.1 Data Flow Diagram:

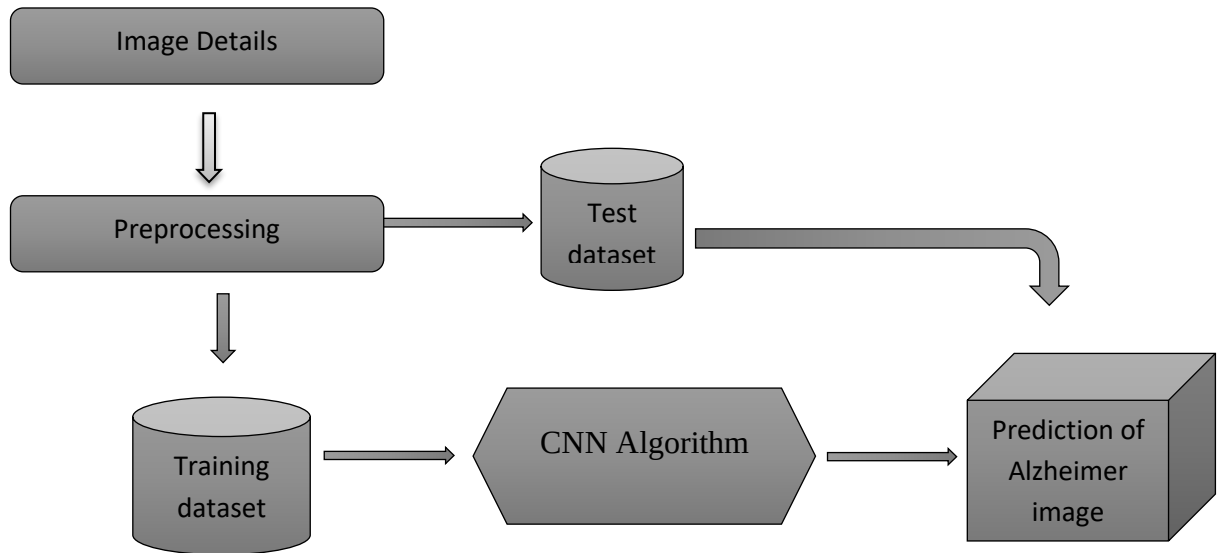


Fig: Process of dataflow diagram

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

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

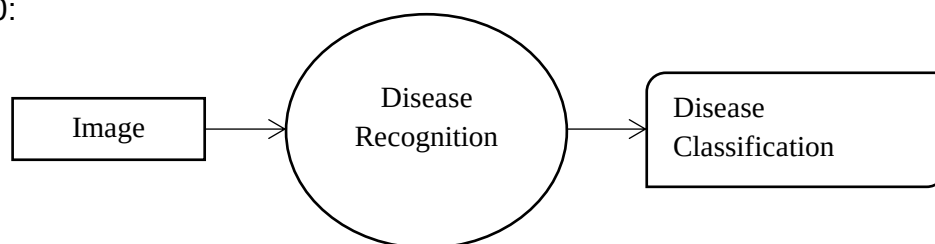
Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

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

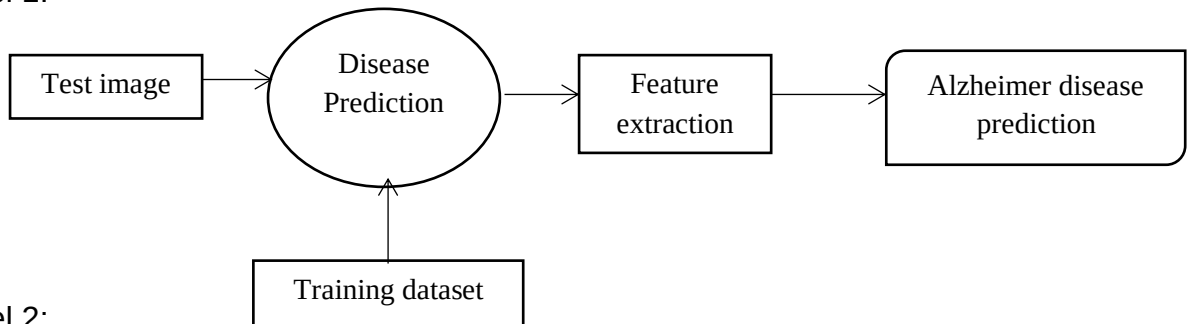
Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

DFD levels and layers A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish. DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

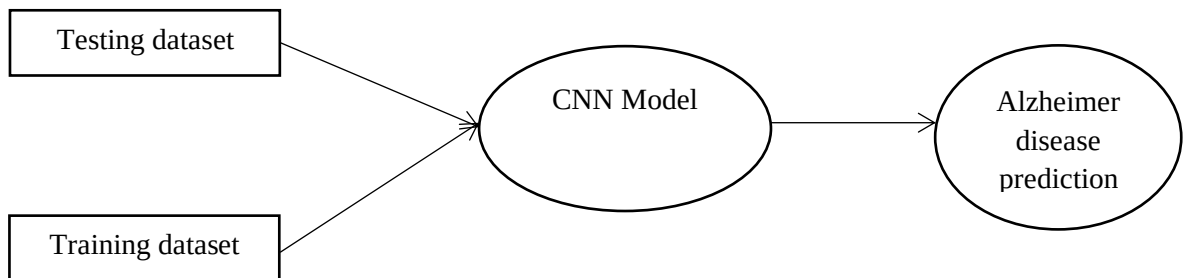
Level 0:



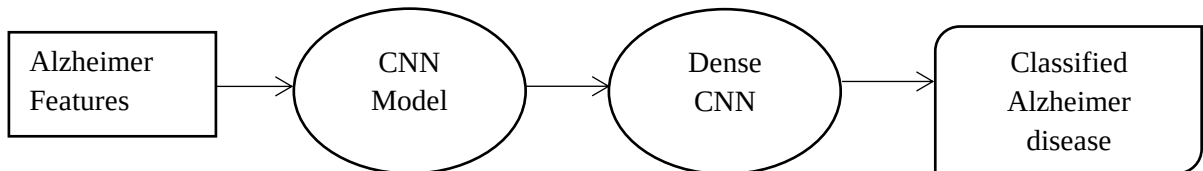
Level 1:



Level 2:

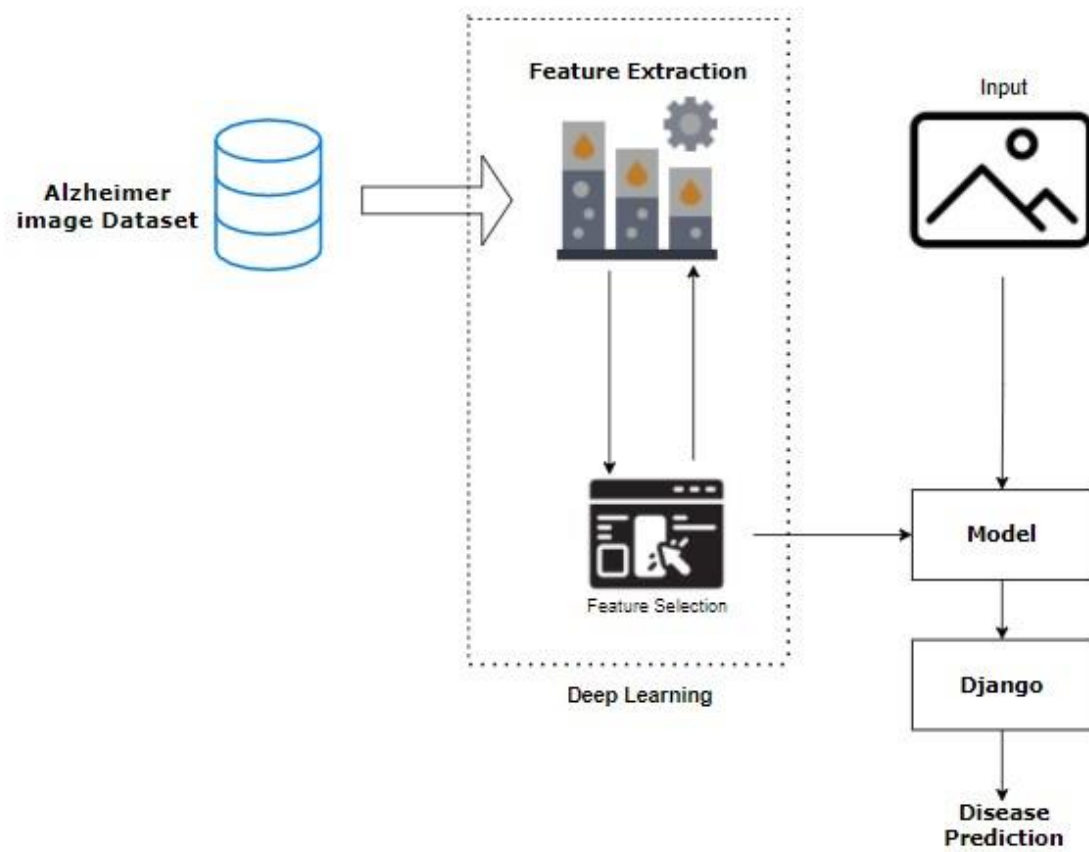


Level 3:



CHAPTER 7

13. SYSTEM ARCHITECTURE:



14.WORKFLOW DIAGRAM:

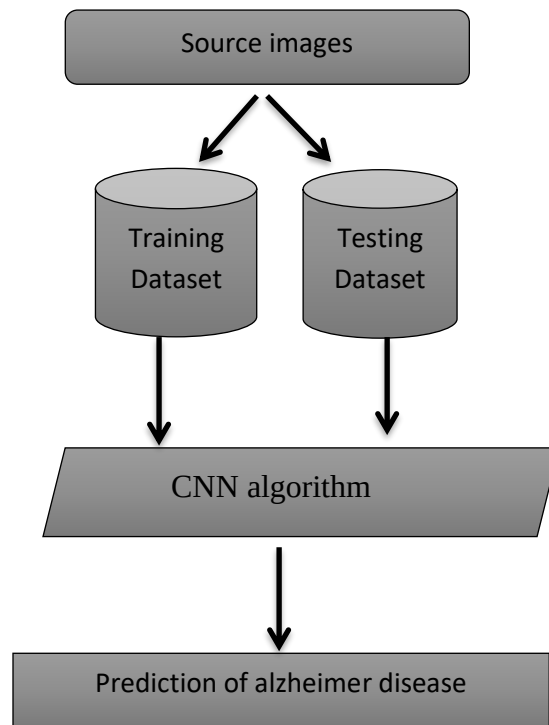
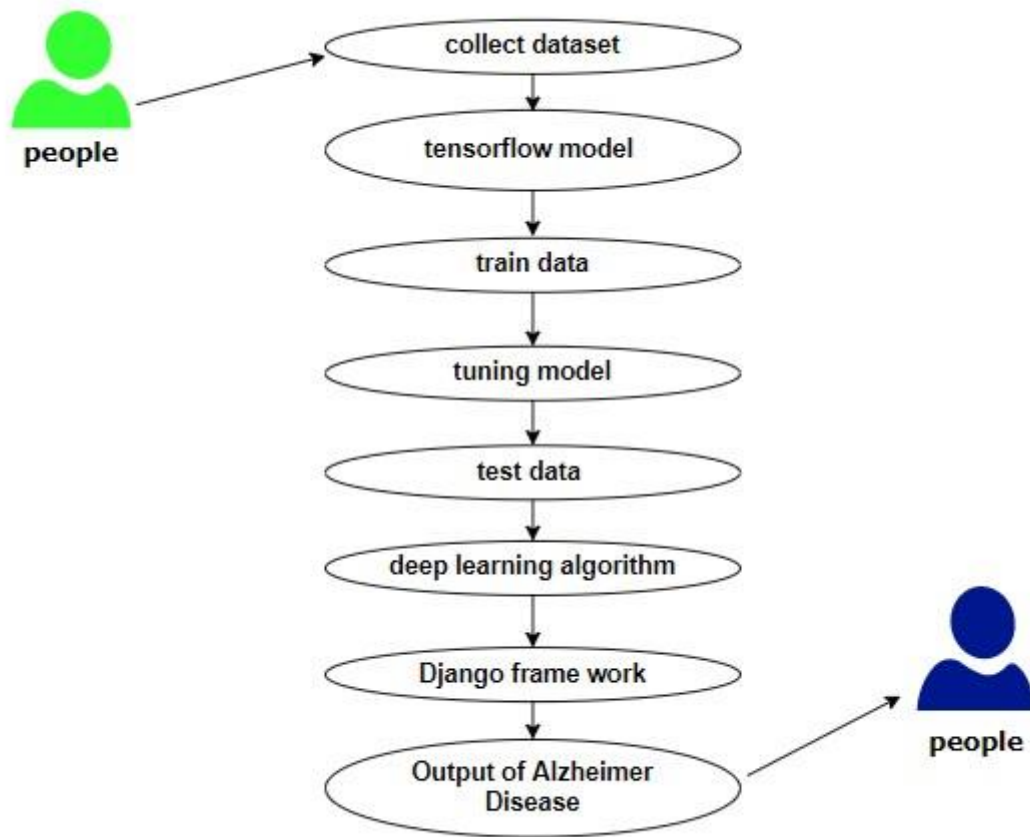


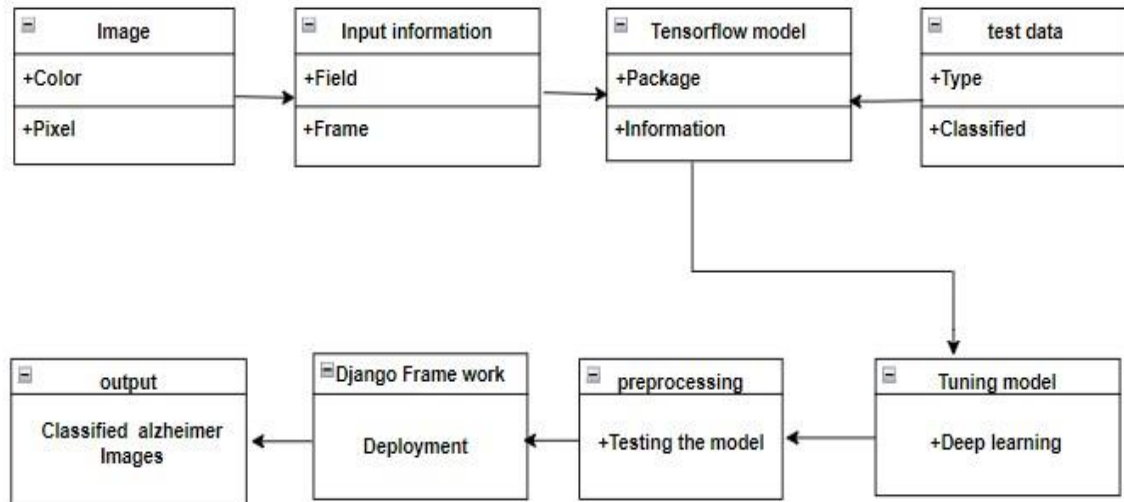
Fig: Workflow Diagram

15. USECASE DIAGRAM:



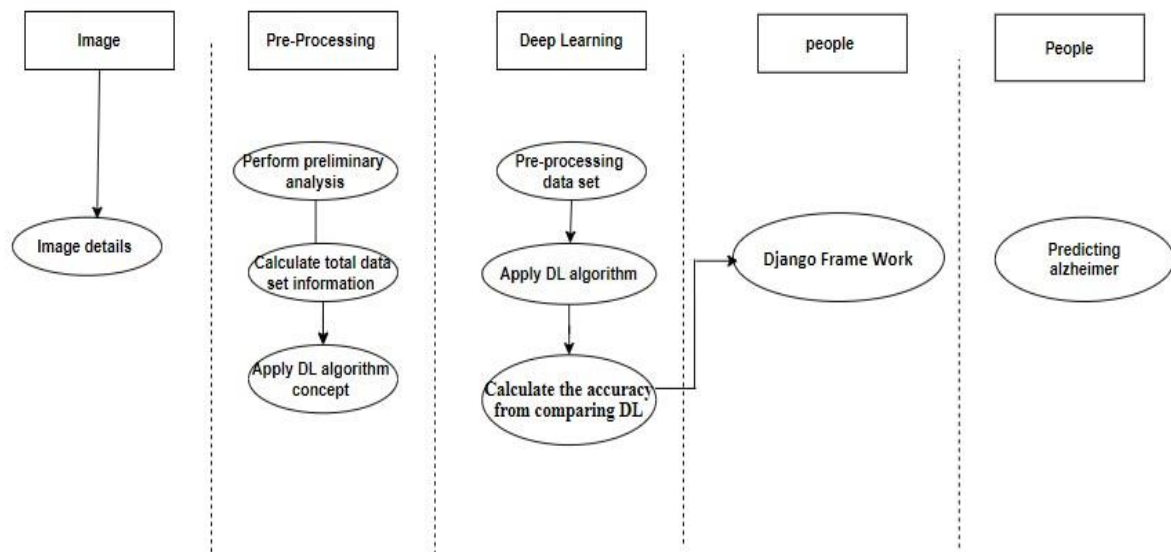
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

16.CLASS DIAGRAM:



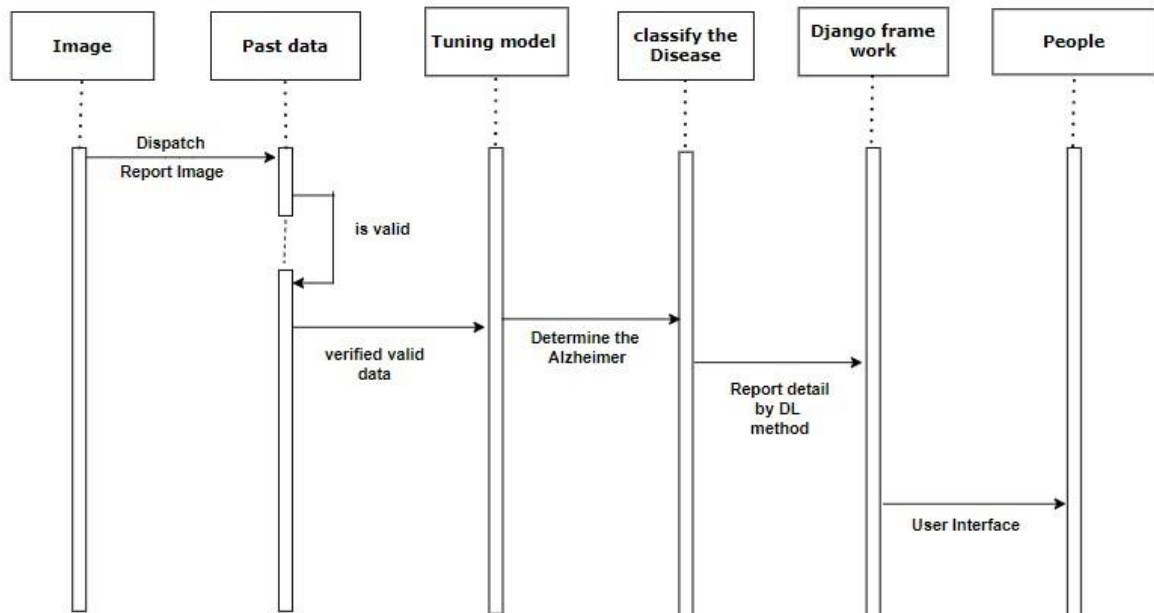
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

17. ACTIVITY DIAGRAM:



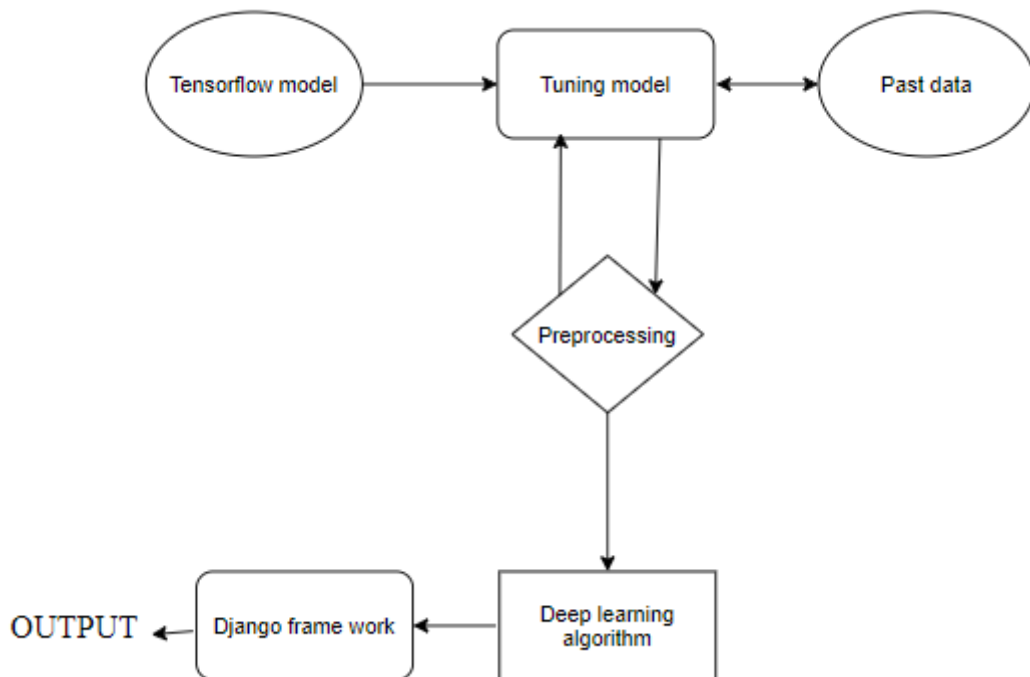
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.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

18. SEQUENCE DIAGRAM:



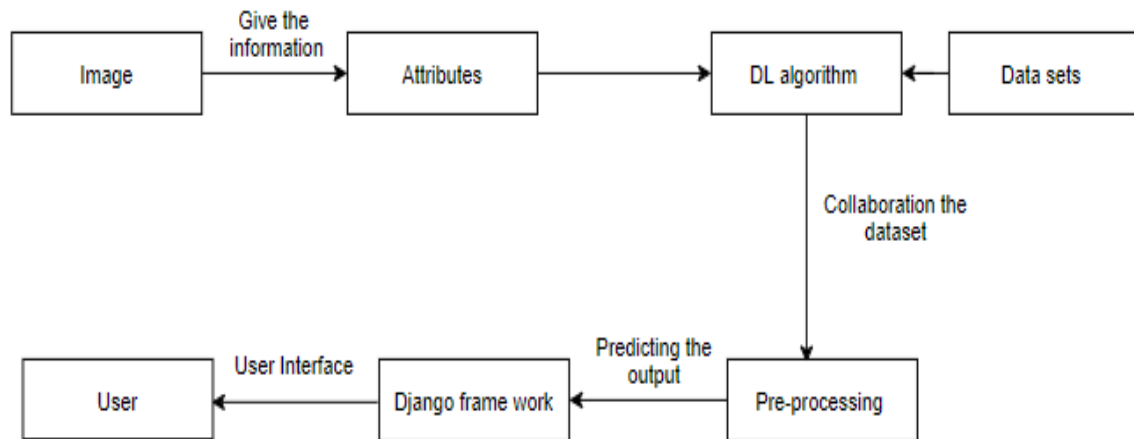
Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying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

19.ER DIAGRAM:



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

20. COLLABORATION DIAGRAM:



A collaboration diagram shows the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction. The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram. It can be the decomposition of a use case, use case diagram, or part of a use case diagram. The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

CHAPTER 8

21. ENVIRONMENTAL REQUIREMENTS:

21.1 Software Requirements:

Operating System : Windows / Linux
Simulation Tool : Anaconda with Jupyter Notebook

21.2 Hardware requirements:

Processor : Pentium IV/III
Hard disk : minimum 80 GB
RAM : minimum 2 GB

22. SOFTWARE DESCRIPTION:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

22.1 ANACONDA NAVIGATOR:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

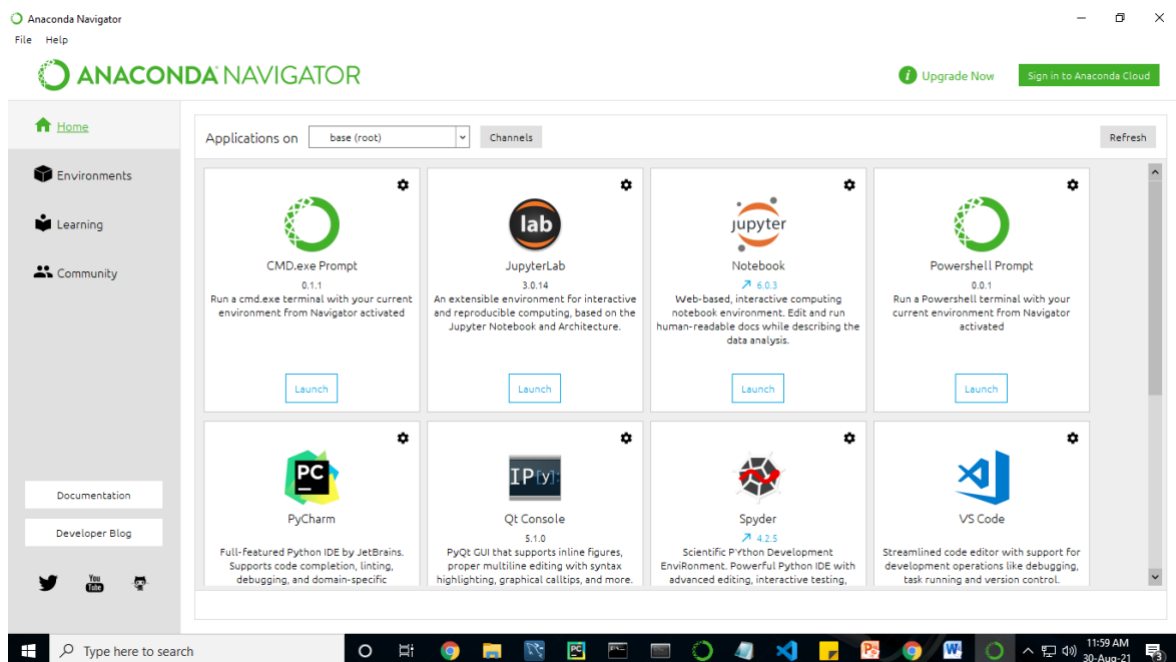
The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

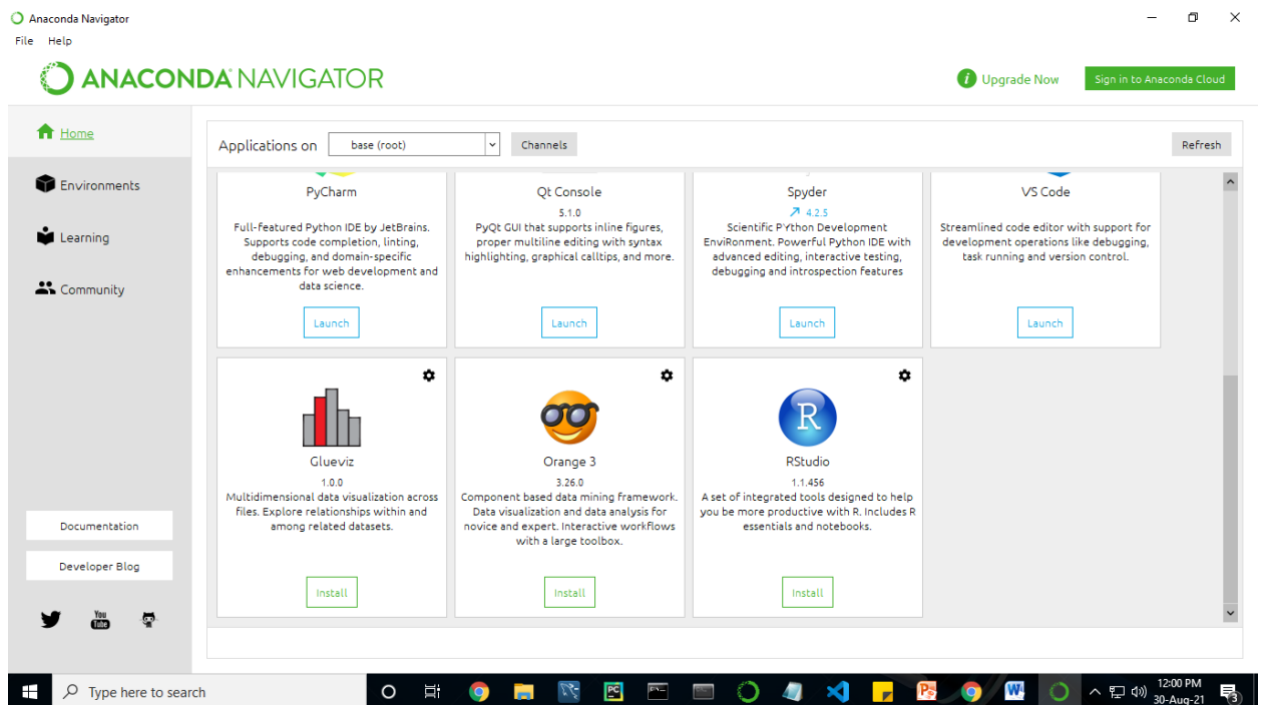
Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab

- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)





Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with `conda list` on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

22.2 JUPYTER NOTEBOOK:

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

Running the Jupyter Notebook

Launching *Jupyter Notebook App*: The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): "jupyter notebook"

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- ❖ Launch the jupyter notebook app
- ❖ In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- ❖ Click on the menu *Help -> User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- ❖ You can run the notebook document step-by-step (one cell a time) by pressing *shift + enter*.
- ❖ You can run the whole notebook in a single step by clicking on the menu *Cell -> Run All*.
- ❖ To restart the kernel (i.e. the computational engine), click on the menu *Kernel -> Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App: The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

kernel: A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist ([official kernels](#)). When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results. Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.

- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

22.3 PYCHARM:

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. Code faster and with more easily in a smart and configurable editor with code completion, snippets, code folding and split windows support.

PyCharm Features

- **Intelligent Coding Assistance** - PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.
- **Intelligent Code Editor** - PyCharm's smart code editor provides first-class support for Python, JavaScript, CoffeeScript, TypeScript, CSS, popular template languages and more. Take advantage of language-aware code completion, error detection, and on-the-fly code fixes!
- **Smart Code Navigation** - Use smart search to jump to any class, file or symbol, or even any IDE action or tool window. It only takes one click to switch to the declaration, super method, test, usages, implementation, and more.
- **Fast and Safe Refactorings** - Refactor your code the intelligent way, with safe Rename and Delete, Extract Method, Introduce Variable, Inline Variable

or Method, and other refactorings. Language and framework-specific refactorings help you perform project-wide changes.

- **Built-in Developer Tools** - PyCharm's huge collection of tools out of the box includes an integrated debugger and test runner; Python profiler; a built-in terminal; integration with major VCS and built-in database tools; remote development capabilities with remote interpreters; an integrated ssh terminal; and integration with Docker and Vagrant.
- **Debugging, Testing and Profiling** - Use the powerful debugger with a graphical UI for Python and JavaScript. Create and run your tests with coding assistance and a GUI-based test runner. Take full control of your code with Python Profiler integration.
- **VCS, Deployment and Remote Development** - Save time with a unified UI for working with Git, SVN, Mercurial or other version control systems. Run and debug your application on remote machines. Easily configure automatic deployment to a remote host or VM and manage your infrastructure with Vagrant and Docker.
- **Database tools** - Access Oracle, SQL Server, PostgreSQL, MySQL and other databases right from the IDE. Rely on PyCharm's help when editing SQL code, running queries, browsing data, and altering schemas.
- **Web Development** - In addition to Python, PyCharm provides first-class support for various Python web development frameworks, specific template languages, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js, and more.
- **Python Web frameworks** - PyCharm offers great framework-specific support for modern web development frameworks such as Django, Flask, Google App Engine, Pyramid, and web2py, including Django templates debugger, manage.py and appcfg.py tools, special autocompletion and navigation, just to name a few.
- **JavaScript & HTML** - PyCharm provides first-class support for JavaScript, CoffeeScript, TypeScript, HTML and CSS, as well as their modern successors. The JavaScript debugger is included in PyCharm and is integrated with the Django server run configuration.
- **Live Edit** - Live Editing Preview lets you open a page in the editor and the browser and see the changes being made in code instantly in the browser.

PyCharm auto-saves your changes, and the browser smartly updates the page on the fly, showing your edits.

- **Scientific Tools** - PyCharm integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.
- **Interactive Python console** - You can run a REPL Python console in PyCharm which offers many advantages over the standard one: on-the-fly syntax check with inspections, braces and quotes matching, and of course code completion.
- **Scientific Stack Support** - PyCharm has built-in support for scientific libraries. It supports Pandas, Numpy, Matplotlib, and other scientific libraries, offering you best-in-class code intelligence, graphs, array viewers and much more.
- **Conda Integration** - Keep your dependencies isolated by having separate Conda environments per project, PyCharm makes it easy for you to create and select the right environment.
- **Customizable and Cross-platform IDE** - Use PyCharm on Windows, Mac OS and Linux with a single license key. Enjoy a fine-tuned workspace with customizable color schemes and key-bindings, with VIM emulation available.
- **Customizable UI** - Enjoy a fine-tuned workspace with customizable color schemes and key-bindings.
- **Plugins** - More than 10 years of IntelliJ platform development gives PyCharm 50+ IDE plugins of different nature, including support for additional VCS, integrations with different tools and frameworks, and editor enhancements such as Vim emulation.
- **Cross-platform IDE** - PyCharm works on Windows, Mac OS or Linux. You can install and run PyCharm on as many machines as you have, and use the same environment and functionality across all your machines.

CHAPTER 9

23. PYTHON:

Introduction:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)) - cite note-32 Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

23.1 History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of

2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

23.2 Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard

library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one— and preferably only one —obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group Monty Python and in occasionally

playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas

23.3 Syntax and Semantics :

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

23.4.Indentation :

Main article: Python syntax and semantics & Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

23.5 Statements and control flow :

Python's statements include:

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.
- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.
- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.

- The assert statement, used during debugging to check for conditions that should apply.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.^{[80][81]} However, better support for co-routine-like functionality is provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

23.6 Expressions :

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating-point division. Python also uses the ** operator for exponentiation.

- From Python 3.5, the new `@` infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.
- From Python 3.8, the syntax `:=`, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
- In Python, `==` compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example `A<=B<=C`.
- Python uses the words `and`, `or`, `not` for its boolean operators rather than the symbolic `&&`, `||`, `!` used in Java and C.
- Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.
- Anonymous functions are implemented using `lambda` expressions; however, these are limited in that the body can only be one expression.
- Conditional expressions in Python are written as `x if c else y` (different in order of operands from the `c ? x : y` operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as `[1, 2, 3]`, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The `+` operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable `t` initially equal to `(1, 2, 3)`, executing `t = t + (4, 5)` first evaluates `t + (4, 5)`, which yields `(1, 2, 3, 4, 5)`, which is then assigned back to `t`, thereby effectively "modifying the contents" of `t`, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.
- Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of

the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

- Python has a "string format" operator `%`. This functions analogously to `printf` format strings in C, e.g. `"spam=%s eggs=%d" % ("blah",2)` evaluates to `"spam=blah eggs=2"`. In Python 3 and 2.6+, this was supplemented by the `format()` method of the `str` class, e.g. `"spam={0} eggs={1}".format("blah",2)`. Python 3.6 added "f-strings": `blah = "blah"; eggs = 2; f"spam={blah} eggs={eggs}"`
- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. `"spam" + "eggs"` returns `"spameggs"`. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. `"2" + "2"` returns `"2"`.
- Python has various kinds of string literals:
 - Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (`\`) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".
 - Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
 - Raw string varieties, denoted by prefixing the string literal with an `r`. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.
- Python has array index and array slicing expressions on lists, denoted as `a[Key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The `eval()` vs. `exec()` built-in functions (in Python 2, `exec` is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as `a=1` cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator `=` for an equality operator `==` in conditions: `if (c==1) {...}` is syntactically valid (but probably unintended) C code but `if c=1: ...` causes a syntax error in Python.

Methods :

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called d-under methods due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as `print`, `length`, `comparison`, support for arithmetic operations, type conversion, and many more.

Typing :

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta-programming and reflection.

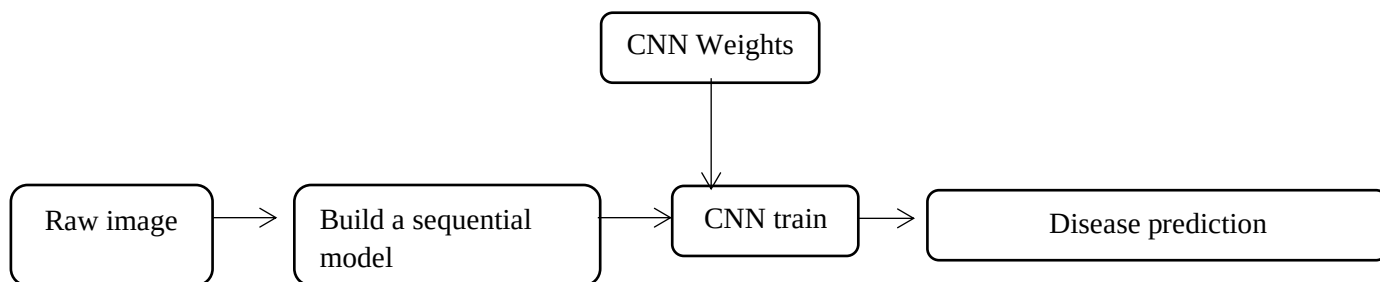
Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython <https://en.wikipedia.org/wiki/CPython>. An experimental optional static type checker named mypy supports compile-time type checking.

CHAPTER 10

24.METHODOLOGY:

Preprocessing and Training the model (CNN): The dataset is preprocessed such as Image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image. A dataset consisting of about images of Alzheimer disease images, out of which any image can be used as a test image for the software.



The train dataset is used to train the model (CNN) so that it can identify the test image and the disease it has. CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the disease if the Alzheimer image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the disease.

24.1 CNN Model steps:

Conv2d:

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

The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are

essentially, the weighted sums (with the weights being the values of the kernel itself) of the input features located roughly in the same location of the output pixel on the input layer.

Whether or not an input feature falls within this “roughly same location”, gets determined directly by whether it’s in the area of the kernel that produced the output or not. This means the size of the kernel directly determines how many (or few) input features get combined in the production of a new output feature.

This is all in pretty stark contrast to a fully connected layer. In the above example, we have $5 \times 5 = 25$ input features, and $3 \times 3 = 9$ output features. If this were a standard fully connected layer, you’d have a weight matrix of $25 \times 9 = 225$ parameters, with every output feature being the weighted sum of every single input feature. Convolutions allow us to do this transformation with only 9 parameters, with each output feature, instead of “looking at” every input feature, only getting to “look” at input features coming from roughly the same location. Do take note of this, as it’ll be critical to our later discussion.

24.2 MaxPooling2D layer

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

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

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

Arguments

- `pool_size`: integer or tuple of 2 integers, window size over which to take the maximum. (2, 2) will take the max value over a 2x2 pooling window. If only one integer is specified, the same window length will be used for both dimensions.
- `strides`: Integer, tuple of 2 integers, or None. Strides values. Specifies how far the pooling window moves for each pooling step. If None, it will default to `pool_size`.
- `padding`: One of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.
- `data_format`: A string, one of `channels_last` (default) or `channels_first`. The ordering of the dimensions in the inputs. `channels_last` corresponds to inputs with shape (batch, height, width, channels) while `channels_first` corresponds to inputs with shape (batch, channels, height, width). It defaults to the `image_data_format` value found in your Keras config file at `~/.keras/keras.json`. If you never set it, then it will be "channels_last".

Input shape

- If `data_format='channels_last'`: 4D tensor with shape (batch_size, rows, cols, channels).
- If `data_format='channels_first'`: 4D tensor with shape (batch_size, channels, rows, cols).

Output shape

- If `data_format='channels_last'`: 4D tensor with shape (batch_size, pooled_rows, pooled_cols, channels).
- If `data_format='channels_first'`: 4D tensor with shape (batch_size, channels, pooled_rows, pooled_cols).

24.3 Flatten layer

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

Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4)

Flatten has one argument as follows

```
keras.layers.Flatten(data_format = None)
```

data_format is an optional argument and it is used to preserve weight ordering when switching from one data format to another data format. It accepts either channels_last or channels_first as value. channels_last is the default one and it identifies the input shape as (batch_size, ..., channels) whereas channels_first identifies the input shape as (batch_size, channels, ...)

24.4 Dense layer

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

Note: If the input to the layer has a rank greater than 2, then Dense computes the dot product between the inputs and the kernel along the last axis of the inputs and axis 0 of the kernel (using `tf.tensordot`). For example, if input has dimensions (batch_size, d0, d1), then we create a kernel with shape (d1, units), and the kernel operates along axis 2 of the input, on every sub-tensor of shape (1, 1, d1) (there are batch_size * d0 such sub-tensors). The output in this case will have shape (batch_size, d0, units).

Besides, layer attributes cannot be modified after the layer has been called once (except the trainable attribute). When a popular kwarg input_shape is passed, then

keras will create an input layer to insert before the current layer. This can be treated equivalent to explicitly defining an InputLayer.

Arguments

- units: Positive integer, dimensionality of the output space.
- activation: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: $a(x) = x$).
- use_bias: Boolean, whether the layer uses a bias vector.
- kernel_initializer: Initializer for the kernel weights matrix.
- bias_initializer: Initializer for the bias vector.
- kernel_regularizer: Regularizer function applied to the kernel weights matrix.
- bias_regularizer: Regularizer function applied to the bias vector.
- activity_regularizer: Regularizer function applied to the output of the layer (its "activation").
- kernel_constraint: Constraint function applied to the kernel weights matrix.
- bias_constraint: Constraint function applied to the bias vector.

Input shape

N-D tensor with shape: (batch_size, ..., input_dim). The most common situation would be a 2D input with shape (batch_size, input_dim).

Output shape

N-D tensor with shape: (batch_size, ..., units). For instance, for a 2D input with shape (batch_size, input_dim), the output would have shape (batch_size, units).

24.5 Dropout layer

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when training is set to True such that no values are dropped during inference. When using `model.fit`, training will be appropriately set to True automatically, and in other contexts, you can set the kwarg explicitly to True when calling the layer.

(This is in contrast to setting `trainable=False` for a Dropout layer. `trainable` does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

Arguments

- **rate**: Float between 0 and 1. Fraction of the input units to drop.
- **noise_shape**: 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape `(batch_size, timesteps, features)` and you want the dropout mask to be the same for all timesteps, you can use `noise_shape=(batch_size, 1, features)`.

seed: A Python integer to use as random seed.

Image Data Generator:

It is that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This Image Data Generator includes all possible orientation of the image.

Training Process:

`train_datagen.flow_from_directory` is the function that is used to prepare data from the `train_dataset` directory. `Target_size` specifies the target size of the image. `Test_datagen.flow_from_directory` is used to prepare test data for the model and all

is similar as above. `fit_generator` is used to fit the data into the model made above, other factors used are `steps_per_epochs` tells us about the number of times the model will execute for the training data.

Epochs:

It tells us the number of times model will be trained in forward and backward pass.

Validation process:

`Validation_data` is used to feed the validation/test data into the model. `Validation_steps` denotes the number of validation/test samples.

25. ARTIFICIAL NEURAL NETWORK:

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets that look like the figure below. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.

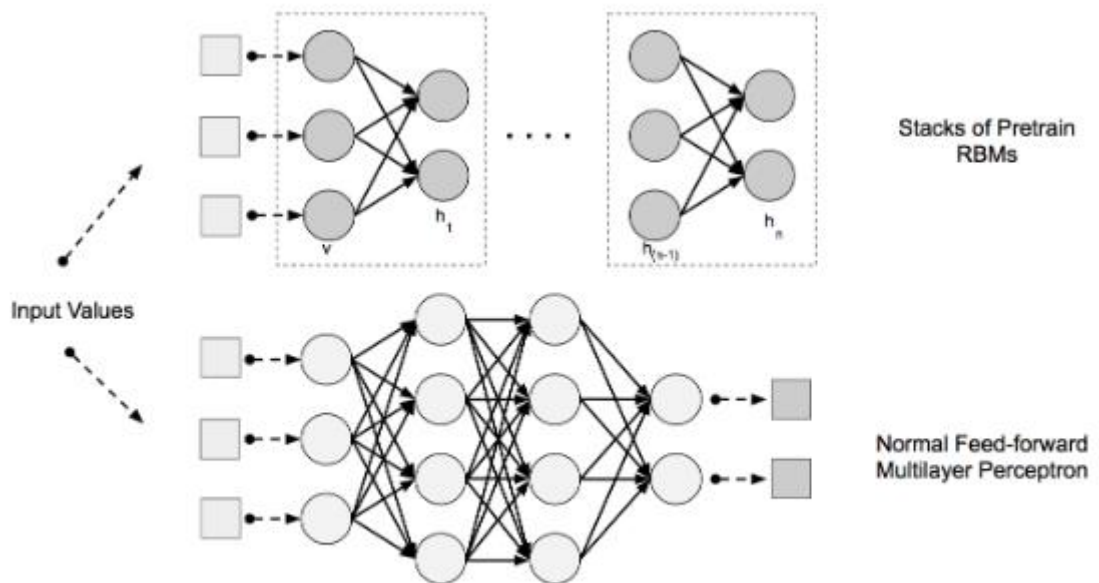
Artificial Neural Network(ANN) uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems.

Key Points related to the architecture:

The network architecture has an input layer, hidden layer (there can be more than 1) and the output layer. It is also called MLP (Multi Layer Perceptron) because of the multiple layers. The hidden layer can be seen as a “distillation layer” that distills some of the important patterns from the inputs and passes it onto the next layer to see. It makes the network faster and efficient by identifying only the important information from the inputs leaving out the redundant information

3. The activation function serves two notable purposes:

- It captures non-linear relationship between the inputs
- It helps convert the input into a more useful output.



Key advantages of neural Networks:

ANNs have some key advantages that make them most suitable for certain problems and situations:

1. ANNs have the ability to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.
2. ANNs can generalize — After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.
3. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables (like how they should be distributed). Additionally, many

studies have shown that ANNs can better model heteroskedasticity i.e. data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in financial time series forecasting (e.g. stock prices) where data volatility is very high.

26. ARCHITECTURE OF CNN:

CONVOLUTIONAL NEURAL NETWORK:

A Convolutional neural network (CNN) is one type of Artificial Neural Network. A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

Models API:

There are three ways to create Keras models:

- The Sequential model, which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away).
- The Functional API, which is an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry strength" model.

- Model subclassing, where you implement everything from scratch on your own. Use this if you have complex, out-of-the-box research use cases.

26.1 Types of Keras Models

Models in keras are available in two types:

- Keras Sequential Model
- Keras Functional API

26.1.1 Sequential Model in Keras

It allows us to create models layer by layer in sequential order. But it does not allow us to create models that have multiple inputs or outputs. It is best for simple stack of layers which have 1 input tensor and 1 output tensor.

This model is not suited when any of the layer in the stack has multiple inputs or outputs. Even if we want non-linear topology, it is not suited.

26.1.2 Functional API in Keras

It provides more flexibility to define a model and add layers in keras. Functional API allows us to create models that have multiple input or output. It also allows us to share these layers. In other words, we can make graphs of layers using Keras functional API.

As functional API is a data structure, it is easy to save it as a single file that helps in recreating the exact model without having the original code. Also its easy to model the graph here and access its nodes as well.

26.2 Model Subclassing in Keras

Sequential model does not allow you much flexibility to create your models. Functional API also only has a little of customization available for you. But you may create your own fully-customizable models in Keras. This is done by subclassing the Model class and implementing a call method.

Input() is used to instantiate a Keras tensor.

A Keras tensor is a symbolic tensor-like object, which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model.

For instance, if a, b and c are Keras tensors, it becomes possible to do: `model = Model(input=[a, b], output=c)`

26.2.1 kernels:

Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are linear transformations, each convolution is a type of affine function.

In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two dimensional matrix) and a 3x3 convolutional kernel (a two dimensional matrix). The kernel strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows. Note, the filter may stride over one or several pixels at a time, this is detailed further below.

In other non-vision applications, a one dimensional convolution may slide vertically over an input matrix.

26.2.2 Padding:

To handle the edge pixels there are several approaches:

- Losing the edge pixels
- Padding with zero value pixels
- Reflection padding

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.

Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasing worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.

With padding, the output from a input of width w and height h would be width w and height h (the same as the input with a single input channel), assuming the kernel takes a stride of one pixel at a time.

Strides:

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1,1), then stride to (1,3), then to (1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one.

With padding, the output from an input of width w , height h and depth 3 would be the ceiling of width $w/2$, height $h/2$ and depth 1, as the kernel outputs a single summed output from each stride.

26.2.3 Rectified Linear Unit (ReLU):

A Rectified Linear Unit is used as a non-linear activation function. A ReLU says if the value is less than zero, round it up to zero.

Normalisation:

Normalisation is the process of subtracting the mean and dividing by the standard deviation. It transforms the range of the data to be between -1 and 1 making the data use the same scale, sometimes called Min-Max scaling.

It is common to normalize the input features, standardising the data by removing the mean and scaling to unit variance. It is often important the input features are centred around zero and have variance in the same order. With some data, such as images the data is scaled so that it's range is between 0 and 1, most simply dividing the pixel values by 255.

Batch normalisation:

Batch normalisation has the benefits of helping to make a network output more stable predictions, reduce overfitting through regularisation and speeds up training by an order of magnitude.

Batch normalisation is the process of carrying normalisation within the scope activation layer of the current batch, subtracting the mean of the batch's activations and dividing by the standard deviation of the batches activations.

This is necessary as even after normalizing the input as some activations can be higher, which can cause the subsequent layers to act abnormally and makes the network less stable.

Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

Importantly, batch normalization works differently during training and during inference.

During training (i.e. when using `fit()` or when calling the layer/model with the argument `training=True`), the layer normalizes its output using the mean and standard deviation of the current batch of inputs. That is to say, for each channel being normalized, the layer returns $\gamma * (\text{batch} - \text{mean}(\text{batch})) / \sqrt{\text{var}(\text{batch}) + \text{epsilon}} + \text{beta}$, where:

- epsilon is small constant (configurable as part of the constructor arguments)
- gamma is a learned scaling factor (initialized as 1), which can be disabled by passing `scale=False` to the constructor.
- beta is a learned offset factor (initialized as 0), which can be disabled by passing `center=False` to the constructor.

During inference (i.e. when using `evaluate()` or `predict()` or when calling the layer/model with the argument `training=False` (which is the default), the layer normalizes its output using a moving average of the mean and standard deviation of the batches it has seen during training. That is to say, it returns $\gamma * (\text{batch} - \text{self.moving_mean}) / \sqrt{\text{self.moving_var} + \text{epsilon}} + \text{beta}$.

`self.moving_mean` and `self.moving_var` are non-trainable variables that are updated each time the layer is called in training mode, as such:

- $\text{moving_mean} = \text{moving_mean} * \text{momentum} + \text{mean}(\text{batch}) * (1 - \text{momentum})$
- $\text{moving_var} = \text{moving_var} * \text{momentum} + \text{var}(\text{batch}) * (1 - \text{momentum})$

As such, the layer will only normalize its inputs during inference *after having been trained on data that has similar statistics as the inference data*.

Arguments

- **axis**: Integer, the axis that should be normalized (typically the features axis). For instance, after a Conv2D layer with `data_format="channels_first"`, set `axis=1` in BatchNormalization.
- **momentum**: Momentum for the moving average.
- **epsilon**: Small float added to variance to avoid dividing by zero.
- **center**: If True, add offset of beta to normalized tensor. If False, beta is ignored.
- **scale**: If True, multiply by gamma. If False, gamma is not used. When the next layer is linear (also e.g. `nn.relu`), this can be disabled since the scaling will be done by the next layer.
- **beta_initializer**: Initializer for the beta weight.
- **gamma_initializer**: Initializer for the gamma weight.
- **moving_mean_initializer**: Initializer for the moving mean.
- **moving_variance_initializer**: Initializer for the moving variance.
- **beta_regularizer**: Optional regularizer for the beta weight.
- **gamma_regularizer**: Optional regularizer for the gamma weight.
- **beta_constraint**: Optional constraint for the beta weight.
- **gamma_constraint**: Optional constraint for the gamma weight.

Call arguments

- **inputs**: Input tensor (of any rank).
- **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode.

27. TYPES OF CNN:

- AlexNet
- LeNet

27.1. ALEXNET:

AlexNet is the name of a convolutional neural network which has had a large impact on the field of machine learning, specifically in the application of deep

learning to machine vision. AlexNet was the first convolutional network which used GPU to boost performance.

AlexNet architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 softmax layer. Each convolutional layer consists of convolutional filters and a nonlinear activation function ReLU. The pooling layers are used to perform max pooling.

Architecture of AlexNet:

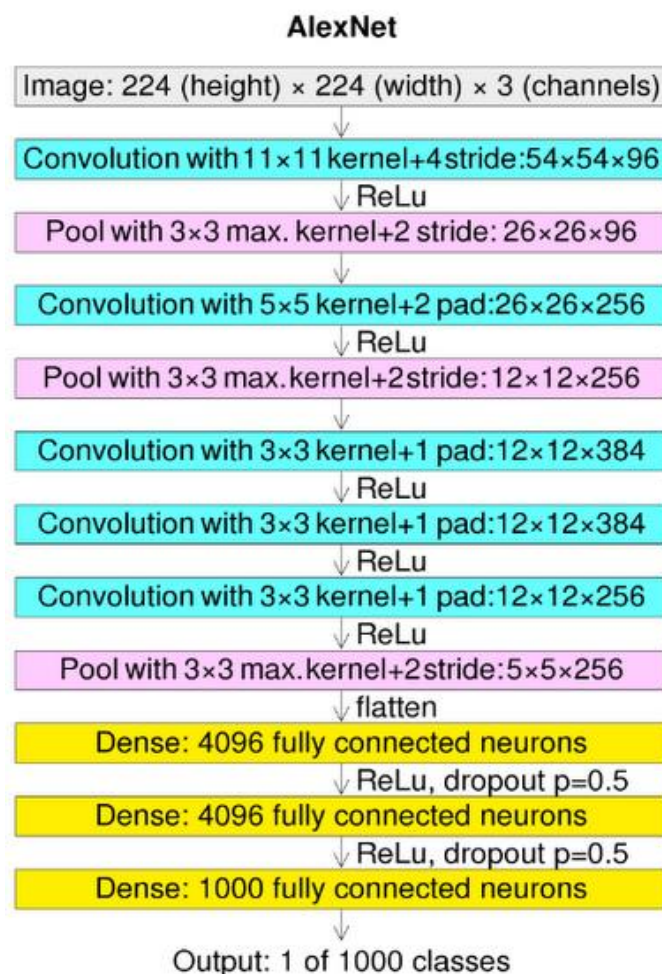


Fig: Architecture of AlexNet

Convolutional layers:

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-

specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Pooling layers:

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

Dense or Fully connected layers:

Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

27.2. LENET:

LeNet was one among the earliest convolutional neural networks which promoted the event of deep learning. After innumerable years of analysis and plenty of compelling iterations, the end result was named LeNet.

Architecture of LeNet-5:

LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers.

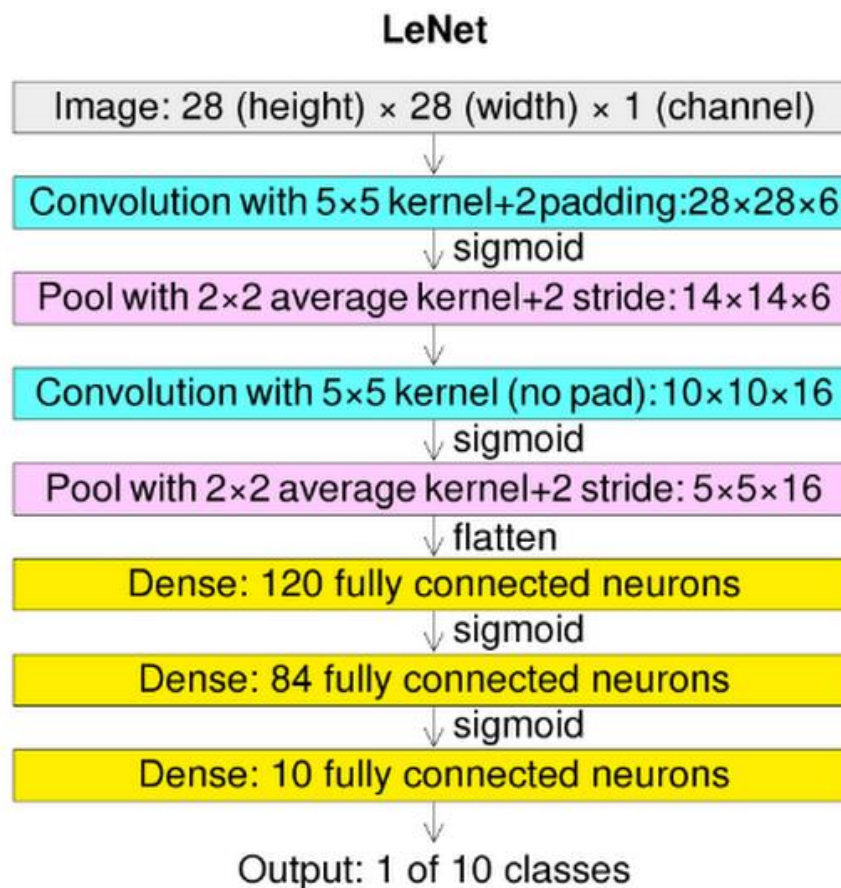


Fig: Architecture of LeNet

Convolutional layers:

Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

Pooling layers:

Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

Dense or Fully connected layers:

Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

28. LIST OF MODULES:

1. Manual Net
2. AlexNet
3. LeNet
4. Deploy

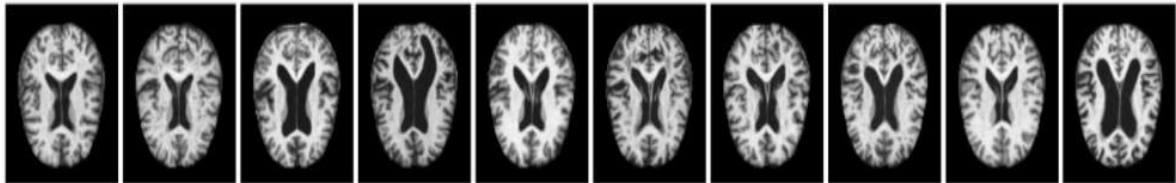
29. MODULE DESCRIPTION:**29.1. IMPORT THE GIVEN IMAGE FROM DATASET:**

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size and class-mode from this function we have to train using our own created network by adding layers of CNN.

DEMENTED

Trained data for Demented:

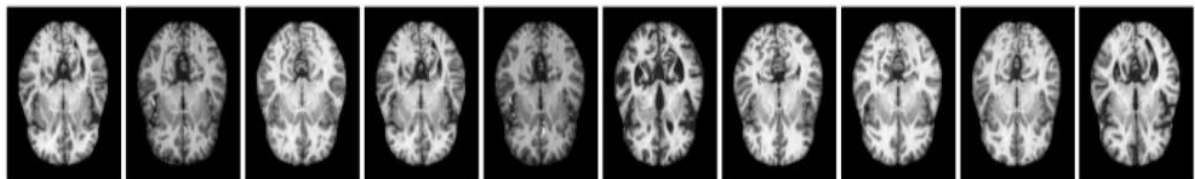
```
===== Images in: Dataset/train/Demented
images_count: 200
min_width: 176
max_width: 208
min_height: 208
max_height: 208
```



NON DEMENTED

Trained data for NonDemented:

```
===== Images in: Dataset/train/NonDemented
images_count: 187
min_width: 176
max_width: 208
min_height: 208
max_height: 208
```



29.2. TO TRAIN THE MODULE BY GIVEN IMAGE DATASET:

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

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 50, 50, 96)	34944
activation_5 (Activation)	(None, 50, 50, 96)	0
max_pooling2d_3 (MaxPooling2D)	(None, 25, 25, 96)	0
conv2d_6 (Conv2D)	(None, 15, 15, 256)	2973952
activation_6 (Activation)	(None, 15, 15, 256)	0
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 256)	0
conv2d_7 (Conv2D)	(None, 5, 5, 384)	885120
activation_7 (Activation)	(None, 5, 5, 384)	0
flatten (Flatten)	(None, 9600)	0
dense (Dense)	(None, 4096)	39325696
activation_8 (Activation)	(None, 4096)	0
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
activation_9 (Activation)	(None, 4096)	0
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 1000)	4097000
activation_10 (Activation)	(None, 1000)	0

Fig: CNN Model Summary details

29.3. WORKING PROCESS OF LAYERS IN CNN MODEL:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a

restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

Input Layer:

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

Convo Layer:

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

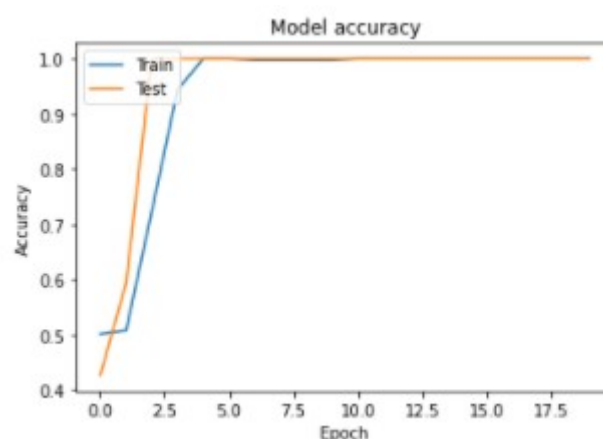


Fig: CNN model trained dataset accuracy

Pooling Layer:

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

Fully Connected Layer (FC):

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

Softmax / Logistic Layer:

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

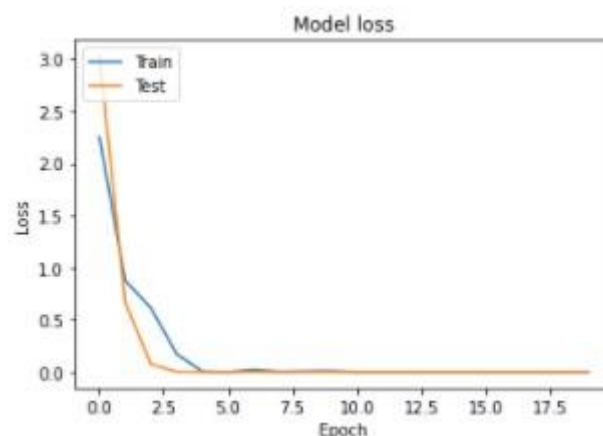


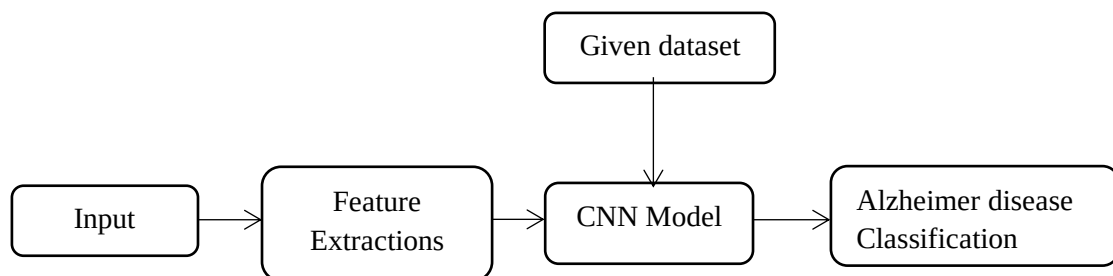
Fig: CNN model trained dataset loss values

Output Layer:

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

29.4. CLASSIFICATION IDENTIFICATION:

We give input image using keras preprocessing package. That input Image converted into array value using pillow and image to array function package. We have already classified disease of Alzheimer in our dataset. It classifies what are the Alzheimer disease. Then we have to predict our Alzheimer diseases using predict function.



The Alzheimer disease recognition method is based on a two-channel architecture that is able to recognize disease of Alzheimer. The Alzheimer disease parts are cropped and extracted and then used as the input into the inception layer of the CNN. The Training phase involves the feature extraction and classification using convolution neural network.

Libraries Required:

- ✓ **tensorflow**: Just to use the tensor board to compare the loss and adam curve our result data or obtained log.
- ✓ **keras**: To pre-process the image dataset.
- ✓ **matplotlib**: To display the result of our predictive outcome.
- ✓ **os**: To access the file system to read the image from the train and test directory from our machines.

TensorFlow:

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Let us now consider the following important features of TensorFlow –

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Keras:

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

Features

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features –

- Consistent, simple and extensible API.

- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

Matplotlib:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPython or Tkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

OS:

The OS module in Python comes with various functions that enables developers to interact with the Operating system that they are currently working on. In this article we'll be learning mainly to create and delete a directory/folder, rename a directory and even basics of file handling.

Python OS module provides the facility to establish the interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

The OS comes under Python's standard utility modules. This module offers a portable way of using operating system dependent functionality.

CHAPTER 11

30. DEPLOY

Deploying the model in Django Framework and predicting output

In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our django framework for providing better user interface and predicting the output whether the given OCT image is CNV / DME / DRUSEN / NORMAL.

30.1 django

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML,

etc). The site you are currently reading is built with Django!

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (see Website security for more details of such attacks).

Scalable

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

30.2 HTML

HTML stands for HyperText Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.



`<!DOCTYPE html>` — This tag **specifies the language** you will write on the page. In this case, the language is HTML 5.

`<html>` — This tag signals that from here on we are going to write in HTML code.

`<head>` — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

`<body>` — This is where the **content of the page** goes.

Further Tags

Inside the `<head>` tag, there is one tag that is always included: `<title>`, but there are others that are just as important:

`<title>`

This is where we **insert the page name** as it will appear at the top of the browser window or tab.

`<meta>`

This is where information *about* the document is stored: character encoding, name (page context), description.

HeadTag

```
<head>
```

```
<title>My First Webpage</title>
```

```
<meta charset="UTF-8">
```

```
<meta name="description" content="This field contains information about your  
page. It is usually around two sentences long.">
```

```
<meta name="author" content="Conor Sheils">
```

```
</header>
```

Adding Content

Next, we will make `<body>` tag.

The HTML `<body>` is where we add the content which is designed for viewing by human eyes.

This includes **text**, **images**, **tables**, **forms** and everything else that we see on the internet each day.

How to Add HTML Headings To Your Web Page

In HTML, headings are written in the following elements:

- `<h1>`
- `<h2>`
- `<h3>`
- `<h4>`
- `<h5>`
- `<h6>`

As you might have guessed `<h1>` and `<h2>` should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Search engine bots use this order when deciphering which information is most important on a page.

Creating Your Heading

Let's try it out. On a new line in the HTML editor, type:

```
<h1>Welcome to My Page</h1>
```

And hit save. We will save this file as "index.html" in a new folder called "my webpage."

***The Moment of Truth:** Click the newly saved file and your first ever web page should open in your default browser. It may not be pretty it's yours... all yours. *Evil laugh**

Add Text In HTML

Adding text to our HTML page is simple using an element opened with the tag `<p>` which **creates a new paragraph**. We place all of our regular text inside the element `<p>`.

Element	Meaning	Purpose
<code></code>	Bold	Highlight important information
<code></code>	Strong	Similarly to bold, to highlight key text
<code><i></code>	Italic	To denote text
<code></code>	Emphasised Text	Usually used as image captions
<code><mark></code>	Marked Text	Highlight the background of the text

<small>	Small Text	To shrink the text
<strike>	Striked Out Text	To place a horizontal line across the text
<u>	Underlined Text	Used for links or text highlights
<ins>	Inserted Text	Displayed with an underline to show an inserted text
<sup>	Superscript Text	Another typographical presentation style

When we write text in HTML, we also have a number of other elements we can use to **control the text or make it appear in a certain way**.

Add Links In HTML

As you may have noticed, the internet is made up of lots of links.

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the **<a>** tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags**.

```
<a href="http://www.google.com">Google</a>
```

Image Tag

In today's modern digital world, images are everything. The **** tag has everything you need to display images on your site. Much like the **<a>** anchor element, **** also contains an attribute.

The attribute *features information* for your computer regarding the **source, height, width** and **alt text** of the image

```

```

30.3 CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.

CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

CSS Syntax

```
Selector {  
  
    Property 1 : value;  
  
    Property 2 : value;  
  
    Property 3 : value;  
  
}
```

For example:

```
h1  
  
{  
  
    Color: red;  
  
    Text-align: center;  
  
}  
  
#unique
```

```
{  
  
    color: green;  
  
}
```

- Selector: selects the element you want to target
- Always remains the same whether we apply internal or external styling
- There are few basic selectors like tags, id's, and classes
- All forms this key-value pair
- Keys: properties(attributes) like color, font-size, background, width, height,etc
- Value: values associated with these properties

CSS Comment

- Comments don't render on the browser
- Helps to understand our code better and makes it readable.
- Helps to debug our code
- Two ways to comment:
 - Single line

CSS How-To

- There are 3 ways to write CSS in our HTML file.
 - Inline CSS
 - Internal CSS
 - External CSS
- Priority order

Inline > Internal > External

Inline CSS

- Before CSS this was the only way to apply styles
- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element
- The idea of separation of concerns was lost
- Example:

```
<h3 style=" color:red"> Have a great day </h3>
```

```
<p style =" color: green"> I did this , I did that </p>
```

Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document
- Example:

```
< style>
```

```
    h1{
```

```
        color:red;
```

```
    }
```

```
</style>
```

```
<h3> Have a great day </h3>
```

External CSS

- With the help of <link> tag in the head tag, we can apply styles
- Reference is added

- File saved with .css extension
- Redundancy is removed
- The idea of separation of concerns is maintained
- Uniquely applied to each document
- Example:

```
<head>

<link rel="stylesheet" type="text/css" href="name of the Css file">

</head>

    h1{

        color:red;    //.css file

    }
```

CSS Selectors

- The selector is used to target elements and apply CSS
- Three simple selectors
 - Element Selector
 - Id Selector
 - Class Selector
- Priority of Selectors

CSS Colors

- There are different colouring schemes in CSS
- **RGB**-This starts with RGB and takes 3 parameter
- **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
- **RGBA**-This starts with RGB and takes 4 parameter

CSS Background

- There are different ways by which CSS can have an effect on HTML elements
- Few of them are as follows:
 - Color – used to set the color of the background
 - Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
 - Image – used to set an image as the background
 - Position – used to determine the position of the image
 - Attachment – It basically helps in controlling the mechanism of scrolling

CSS BoxModel

- Every element in CSS can be represented using the BOX model
- It allows us to add a border and define space between the content
- It helps the developer to develop and manipulate the elements
- It consists of 4 edges
 - Content edge – It comprises of the actual content
 - Padding edge – It lies in between content and border edge
 - Border edge – Padding is followed by the border edge
 - Margin edge – It is an outside border and controls the margin of the element.

CHAPTER 12

REFERENCES

1. Ali, E.M., Seddik, A.F., Haggag, M.H.: Automatic detection and classification of alzheimer's disease from mri using tannn. *International Journal of Computer Applications* 148(9) (2016)
2. Arvesen, E.: Automatic Classification of Alzheimers Disease from Structural MRI. Master's thesis (2015)
3. Brosch, T., Tam, R., Initiative, A.D.N.: Manifold learning of brain mris by deep learning. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. pp. 633–640. Springer (2013)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. pp. 248–255. IEEE (2009)
5. Grady, C.L., McIntosh, A.R., Beig, S., Keightley, M.L., Burian, H., Black, S.E.: Evidence from functional neuroimaging of a compensatory prefrontal network in alzheimer's disease. *Journal of Neuroscience* 23(3), 986–993 (2003)

CHAPTER 13

APPENDIX

31. CODING:

To build a model for training and testing:

31.1 Module 1:

```
import os
import numpy as np # linear algebra
import matplotlib.pyplot as plt

In []:
```



```
# DL framework - tensorflow, keras a backend
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,
BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D, MaxPool2D,
LeakyReLU, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
```

```

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense

```

In []:

```

import warnings
warnings.filterwarnings('ignore')

```

In []:

```

dir_name_train_Demented = 'Dataset/train/Demented'
dir_name_train_NonDemented = 'Dataset/train/NonDemented'

```

In []:

```

def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)
    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

    plt.figure(figsize=(35, 10))
    for idx, img_path in enumerate(item_files):
        plt.subplot(2, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')

    plt.tight_layout()

```

In []:

```

def Images_details_Print_data(data, path):
    print(" ===== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.*", recursive=True)]

```

```

data = {}
data['images_count'] = len(files)
data['min_width'] = 10**100 # No image will be bigger than that
data['max_width'] = 0
data['min_height'] = 10**100 # No image will be bigger than that
data['max_height'] = 0

```

for f in files:

```

    im = Image.open(f)
    width, height = im.size
    data['min_width'] = min(width, data['min_width'])
    data['max_width'] = max(width, data['max_width'])
    data['min_height'] = min(height, data['min_height'])
    data['max_height'] = max(height, data['max_height'])

```

```

Images_details_Print_data(data, path)

```

In []:

```

print("")
print("Trained data for Demented:")
print("")
Images_details(dir_name_train_Demented)
print("")
plot_images(dir_name_train_Demented, 10)

```

In []:

```

print("")
print("Trained data for NonDemented:")
print("")
Images_details(dir_name_train_NonDemented)
print("")
plot_images(dir_name_train_NonDemented, 10)

```

In []:

```

Classifier=Sequential()
Classifier.add(Convolution2D(32,(3,3),input_shape=(208,208,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(38, activation='relu'))

```

In []:

```

Classifier.add(Dense(2, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])

```

In []:

```

train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)

```

In []:

```

training_set=train_datagen.flow_from_directory('dataset/Train',target_size=(208,208),batch_size=32,class_mode='categorical')
test_set=test_datagen.flow_from_directory('dataset/Test',target_size=(208,208),batch_size=32,class_mode='categorical')

```

In []:

```

img_dims = 150
epochs = 10
batch_size = 32

```

In []:

Fitting the model

```

history = Classifier.fit_generator(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set,validation_steps=test_set.samples // batch_size)

```

In []:

def graph():

#Plot training & validation accuracy values

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

Plot training & validation loss values

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

In []:

graph()

31.2 Module 2:

DL framework - tensorflow, keras a backend

```
import tensorflow as tf
```

In []:

```
import tensorflow.keras.backend as K
```

In []:

```
from tensorflow.keras.models import Model
```

In []:

```
from tensorflow.keras.models import Sequential
```

In []:

```

from tensorflow.keras.layers import Input
In [:
from tensorflow.keras.layers import Dense
In [:
from tensorflow.keras.layers import Flatten
In [:
from tensorflow.keras.layers import Conv2D
In [:
from tensorflow.keras.layers import MaxPooling2D
In [:
from tensorflow.keras.layers import Dropout
In [:
from tensorflow.keras.layers import LeakyReLU
In [:
from tensorflow.keras.layers import Activation
In [:
from tensorflow.keras.optimizers import Adam
In [:
from tensorflow.keras.preprocessing.image import ImageDataGenerator
In [:
from tensorflow.keras.callbacks import ModelCheckpoint
In [:
from tensorflow.keras.callbacks import ReduceLROnPlateau
In [:
from tensorflow.keras.callbacks import EarlyStopping
In [:
import warnings
warnings.filterwarnings('ignore')
In [:

model = Sequential()

```

```

# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(208,208,3), kernel_size=(11,11),
strides=(4,4), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# 2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(11,11), strides=(1,1),
padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# 3rd Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid'))
model.add(Activation('relu'))
# Passing it to a Fully Connected layer
model.add(Flatten())
# 1st Fully Connected Layer
model.add(Dense(4096, input_shape=(300*300*3,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))
# 2nd Fully Connected Layer
model.add(Dense(4096))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# 3rd Fully Connected Layer
model.add(Dense(1000))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# Output Layer
model.add(Dense(2))

```

```
model.add(Activation('softmax'))
model.summary()
```

```
# Compile the model
```

```
model.compile(loss = 'categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

In []:

```
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In []:

```
training_set=train_datagen.flow_from_directory('dataset/Train', target_size=(208,208), batch_size=32, class_mode='categorical')
test_set=test_datagen.flow_from_directory('dataset/Test', target_size=(208,208), batch_size=32, class_mode='categorical')
```

In []:

```
img_dims = 150
epochs = 20
batch_size = 32
```

In []:

```
#### Fitting the model
```

```
history = model.fit(
    training_set, steps_per_epoch=training_set.samples // batch_size,
    epochs=epochs,
    validation_data=test_set, validation_steps=test_set.samples // batch_size)
```

In []:

```
import matplotlib.pyplot as plt
```

In []:

```
def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
```

```
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

In []:

```
graph()
```

In []:

```
print("[INFO] Calculating model accuracy")
scores = model.evaluate(test_set)
print(f"Test Accuracy: {scores[1]*100}")
```

31.3 Module 3:

```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping
```

In []:

```
from tensorflow.keras.models import Sequential
```

In []:

```
from tensorflow.keras.layers import Convolution2D
```

In []:

```
from tensorflow.keras.layers import MaxPooling2D
```

In []:

```
from tensorflow.keras.layers import Flatten
```

In []:

```
from tensorflow.keras.layers import Dense
```

In []:

```
import warnings
warnings.filterwarnings('ignore')
```

In []:

```
Classifier=Sequential()
```

In []:

```
Classifier.add(Convolution2D(32,3,3,input_shape=(208,208,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Convolution2D(128,3,3,activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(256, activation='relu'))
Classifier.add(Dense(2, activation='softmax'))

Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
Classifier.summary()
```

In []:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In []:

```
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
```

In []:

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In []:

```
training_set=train_datagen.flow_from_directory('dataset/Train',target_size=(208,208),batch_size=32,class_mode='categorical')
```

In []:

```
test_set=test_datagen.flow_from_directory('dataset/Test',target_size=(208,208),batch_size=32,class_mode='categorical')
```

In []:

```
from IPython.display import display
```

In []:

```
img_dims = 150  
epochs = 20  
batch_size = 32
```

In []:

```
#pip install Pillow
```

In []:

```
from PIL import Image
```

In []:

```
Classifier.fit_generator( training_set, steps_per_epoch=training_set.samples //  
batch_size,  
epochs=epochs,  
validation_data=test_set,validation_steps=test_set.samples // batch_size)
```

In []:

```
import h5py
```

In []:

```
Classifier.save('alz_himer5.h5')
```

In []:

```
from keras.models import load_model
```

In []:

```
model=load_model('alz_himer5.h5')
```

In []:

```
import numpy as np
```

In []:

```
from tensorflow.keras.preprocessing import image  
test_image=image.load_img('11.jpg',target_size=(208,208))
```

In []:

```
import matplotlib.pyplot as plt  
img = plt.imshow(test_image)
```

In []:

```
test_image=image.img_to_array(test_image)
```

In []:

```
test_image=np.expand_dims(test_image,axis=0)
```

In []:

```
result=model.predict(test_image)
```

In []:

```
prediction = result[0]
```

In []:

```
classes=training_set.class_indices
```

In []:

```
classes
```

In []:

```
prediction=list(prediction)
```

In []:

```
prediction
```

In []:

```
classes=['Demented', 'NonDemented']
```

In []:

```
output=zip(classes,prediction)
```

In []:

```
output=dict(output)
```

In []:

```
output
```

In []:

```
if output['Demented']==1.0 :  
    print("Demented")  
elif output['NonDemented']==1.0:  
    print("NonDemented")
```

PyCharm:

Views.py

```
from django.shortcuts import render
```

```
from django.http import HttpResponseRedirect  
from django.urls import reverse_lazy  
from django.views.generic import TemplateView  
from employee.forms import EmployeeForm
```

```
from django.views.generic import DetailView  
from employee.models import Employee
```

```
class EmployeeImage(TemplateView):  
    form = EmployeeForm  
    template_name = 'emp_image.html'  
  
    def post(self, request, *args, **kwargs):  
        form = EmployeeForm(request.POST, request.FILES)  
  
        if form.is_valid():
```

```

    obj = form.save()

    return HttpResponseRedirect(reverse_lazy('emp_image_display',
kwargs={'pk': obj.id}))

    context = self.get_context_data(form=form)
    return self.render_to_response(context)

def get(self, request, *args, **kwargs):
    return self.post(request, *args, **kwargs)

class EmplImageDisplay(DetailView):
    model = Employee
    template_name = 'emp_image_display.html'
    context_object_name = 'emp'

def alzemer(request):
    result1 = Employee.objects.latest('id')
    import numpy as np
    import tensorflow as tf
    from tensorflow import keras
    import h5py
    test_image = image.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis=0)
    result = models.predict(test_image)
    prediction = result[0]
    prediction = list(prediction)
    classes=['Demented', 'NonDemented']
    output = zip(classes, prediction)
    output = dict(output)
    if output['Demented'] == 1.0:
        a = "Demented"

```

```
elif output['NonDemented'] == 1.0:  
    a = "NonDemented"
```

emp_image.html

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <title>IMAGE UPLOAD</title>  
</head>  
<style>  
    label  
{  
font-size: 20px;  
color:red;  
font-family: Algerian;  
}  
body  
{  
    background: url(../static/image/paral.gif);  
    background-repeat: no-repeat;  
    background-position: center;  
    background-size: 100% 100%;  
    min-height: 100vh;  
}  
<!-- button-->  
<!--{-->  
<!--font-size: 20px;-->  
<!--font-family: wide latin;-->  
<!--color:black;-->  
<!--background-color: green;-->  
<!-- box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;-->  
<!--}-->  
.button {
```

```

background-color: #4CAF50; /* Green */
border: none;
color: white;
padding: 16px 32px;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 16px;
margin: 4px 2px;
transition-duration: 0.4s;
cursor: pointer;
}
.button1 {
background-color: white;
color: black;
border: 2px solid #4CAF50;
}
#ss
{
font-size: 20px;
color:#20fc03;
background-color: black;
font-family: Times new roman;
}
.button1:hover {
background-color: #4CAF50;
color: white;
}

h2
{
font-size: 20px;
color:black;
background-color: white;

```

```
font-family: Times new roman;
}
```

```
a
```

```
{
font-size: 20px;
color:yellow;
font-family: Times new roman;
}
```

```
.alerts-border {
border: 4px #ff0000 dashed;

animation: blink 0.2s;
animation-iteration-count: infinite;
}
```

```
@keyframes blink { 50% { border-color:yellow ; }
}
```

```
button
```

```
{
font-family: Algerian;
font-size: 35px;
font-weight: bold;
}
```

```
.alerts-border
{
width:500px;
margin-top:7%;
margin-left:28%;
padding: 10px 10px 10px 10px;
```

```
</style>
```

```
<body style="background-color: lightblue;">
```

```
<center><h2 class="blink_me">ALZHEIMER PREDICTION USING DEEP
LEARNING ALGORITHM</h2></center><br><br><br><br>
```



```

background-size: cover;
-webkit-background-size: cover;
-moz-background-size: cover;
-o-background-size: cover;
min-height: 100vh;
}
button
{
font-size: 20px;
font-family: wide latin;
color:black;
background-color: green;
box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;
}
#ss
{
font-size: 20px;
color:#20fc03;
background-color: black;
font-family: Times new roman;
}

h2
{
font-size: 20px;
color:black;
background-color: white;
font-family: Times new roman;
}
a
{
font-size: 20px;
color:#FA2204 ;
font-family: Times new roman;

```

```

}
.alerts-border {
    border: 4px #ff0000 dashed;

    animation: blink 0.2s;
    animation-iteration-count: infinite;
}

@keyframes blink { 50% { border-color:yellow ; }
}
</style>
<body>
<center>
    
    <br>
    <a href="{% url 'al' %}">Result</a>&#160;&#160;&#160;

    <a href="{% url 'hoem' %}">Go Back!!!</a>

</center>
</body>
</html>

```

result.html

```

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>ALZHEIMER PREDICTION OUTPUT RESULT</title>
</head>
<style>

```

```

label
{
font-size: 20px;
color:red;
font-family: Algerian;
}
body
{
background: url(../static/image/paral.gif);
background-repeat: no-repeat;
background-position: center;
background-size: cover;
-webkit-background-size: cover;
-moz-background-size: cover;
-o-background-size: cover;
min-height: 100vh;
}
button
{
font-size: 20px;
font-family: wide latin;
color:black;
background-color: green;
box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;
}
#ss
{
font-size: 20px;
color:#20fc03;
background-color: black;
font-family: Times new roman;
}
.blink_me {
animation: blinker 2s linear infinite;

```

```
}
```

```
@keyframes blinker {
```

```
  50% {
```

```
    opacity: 0;
```

```
  }
```

```
}
```

```
h2
```

```
{
```

```
background-color: #101010;
```

```
font-family: Algerian;
```

```
font-size: 33px;
```

```
letter-spacing: 3px;
```

```
color:#56ff00;
```

```
}
```

```
h1,h3
```

```
{
```

```
font-family: Algerian;
```

```
font-size: 33px;
```

```
letter-spacing: 3px;
```

```
color:red;
```

```
}
```

```
a
```

```
{
```

```
font-size: 20px;
```

```
color:red ;
```

```
font-family: Times new roman;
```

```
}
```

```
</style>
```

```
<body>
```

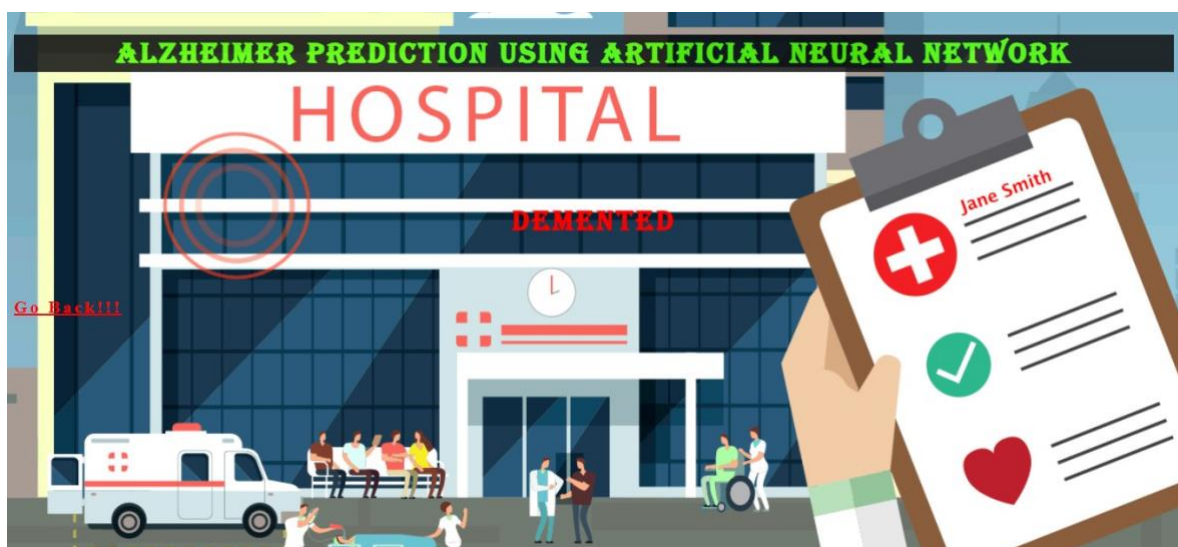
```
<center><h2 class="blink_me">ALZHEIMER PREDICTION USING ARTIFICIAL
```

```

NEURAL NETWORK</h2></center><br><br><br><br>
<marquee
direction="down"><center><b><h1>{{out}}</h1></b></center></marquee>
<h3><a href="{% url 'home' %}">Go Back!!!</a></h3>
</body>
</html>

```

Output Screenshot:



CHAPTER 14

32. Conclusion:

It focused how image from given dataset (trained dataset) and past data set used to predict the pattern of alzheimer diseases using CNN model. This brings some of the following insights about alzheimer disease prediction. The major benefit of the CNN classification framework is the ability to classify images automatically. Alzheimer's disease is a brain disorder that slowly destroys memory and thinking skills and often can't be remedied because the patients are diagnosed too late with the diseases. In this study, we have discussed the overview of methodologies for detecting the abnormalities in alzheimer images which includes collection of alzheimer image data set, preprocessing techniques, feature extraction techniques and classification schemes.

33. Future Work:

- Medical department wants to automate the detecting of alzheimer disease from eligibility process (real time).
- To optimize the work to implement in Artificial Intelligence environment.