**EARLY DIAGNOSIS OF ALZHEMIER’S DISEASE USING DEEP LEARNING**

A PROJECT REPORT

***Submitted by***

BL.EN.U4ECE16205 K.V.JAGADISH KRISHNA

BL.EN.U4ECE16309 C.SAI ABHINAV

BL.EN.U4ECE16332 NARESH KUMAAR A.S.J.

***In partial fulfilment for the award of the degree***

***Of***

**BACHELOR OF TECHNOLOGY**

IN

ELECTRONICS AND COMMUNICATION ENGINEERING



AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

**BANGALORE 560 035**

JUNE-2020

**AMRITA VISHWA VIDYAPEETHAM**

**AMRITA SCHOOL OF ENGINEERING, BANGALORE-560035**



**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “**EARLY DIAGNOSIS OF ALZHEMIER’S DISEASE USING DEEP LEARNING**” submitted

**By**

BL.EN.U4ECE16205         K.V.JAGADISH KRISHNA

      BL.EN.U4ECE16309         C.SAI ABHINAV

      BL.EN.U4ECE16332         NARESH KUMAAR A.S.J.

In partial fulfilment of the requirements of the award of the **Degree in Bachelor of Technology** in “**Electronics and Communication Engineering”** is a Bonafide record of the work carried out under the guidance and supervision ~~at~~ **Amrita School of Engineering, Bangalore.**

**GUIDE**

Ms. Sunitha

Assistant Professor

Department of ECE

Amrita School of Engineering, Bangalore

EXAMINER I                                                                              EXAMINER II

Ms. Lalitha     Ms. Bhavana

Asst. Professor Asst. Professor

**ACKNOWLEDGEMENT**

The culmination of engineering is based entirely on the successful completion of end semester project. It allowed us to explore our interests and be innovative. During this journey I have many people to thank.

We would like to express our gratitude to all who have helped us directly or indirectly in this project. To start with, we thank the Almighty for all his blessings during the tenure of our project.

Firstly, we would like to offer pranam to the lotus feet of Universal guru,**MATA AMRITANANDAMAYI DEVI** who has been a driving force and a source of inspiration for all educational activities and blessed us with her grace to make this a successful project.

I would like to extend my sincere thanks to our Associate Dean **Dr.Rakesh S. G**, Amrita School of Engineering, Bangalore for inspiring us and I am highly indebted to **Dr. N. S. Murthy**, Chairperson, Department of Electronics and Communication Engineering for motivating us towards the successful completion of the project.

We would like to express sincere thanks to our project guide **Ms.Sunitha**, Assistant Professor (Sr. Gr.), Department of Electronics and Communication Engineering who has been very patient and helpful at all times and for pointing us in the right direction.

We also express our sincere gratitude to all the panel members **Ms. Lalitha**, Assistant Professor, Department of Electronics and Communication Engineering and **Ms.Bhavana**, Assistant Professor, Department of Electronics and Communication Engineering for their valuable suggestions and encouragement.

A special thanks to all staff members of the Department of Electronics and Communication Engineering for their support and guidance. We shall remember fondly the whole-hearted support of my family throughout the progress of the project.

**Abstract**

Alzheimer ’s disease is a progressive and irreversible neurological disease and is the most common cause of Dementia in people of the age 65 years and above. Detection of Alzheimer’s disease at prodromal stage is very important as it can prevent serious damage to the patient’s brain.

Alzheimer’s is one of the most common cause of Dementia. Dementia means loss of Cognitive functioning – thinking, remembering and reasoning – and behavioural ability to such an extent that it interferes with Daily life. The image processing is widely used in medical field in order to detect disease and help doctor in decision making based on observation.

The image processing technique to process the Magnetic Resonance Imaging (MRI) of brain from axial plane, coronal plane and sagittal plane. The image segmentation is used to highlight the affected region in brain MRI. The diagnosed region in brain MRI include hippocampus and volume of brain.

In this project, a method to detect Alzheimer’s Disease from MRI using Machine Learning approach is proposed. The proposed approach extracts texture and shape features of the Hippocampus region from the MRI scans and a Neural Network is used as Multi-Class Classifier for detection of various stages of Alzheimer’s Disease. The proposed approach is under implementation and is expected to give better accuracy as compared to conventional approaches.

**Keywords** — Alzheimer’s Disease, Magnetic Resonance Imaging, hippocampus, Moment Invariant, Gray Level Co-occurrence Matrix.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **1.** | **INTRODUCTION** | **1-2** |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **2.** | **LITERATURE SURVEY** | **3-8** |
| **3.** | **METHODOLOGY** | **9-46** |
|  | 3.1 INTRODUCTION TO PYTHON |  |
|  | 3.2 AI AND ML  3.3 ARTIFICIAL NEURAL NETWORK  3.4 CONVOLUTION NEURAL NETWORK  3.5 OPEN CV |  |
| **4.** | **SYSTEM ARCHITECTURE AND DESIGN** | **47-54** |
|  | 4.1 IMAGE ACQUISITION AND PRE-PROCESSING |  |
|  | 4.2 DATA PREPARATION AND MODEL CONSTRUCTION  4.3 MODEL TRAINING  4.4 MODEL TESTING AND EVALUATION |  |
| **5.** | **RESULT AND CONCLUSION** | **55-60** |
|  | 5.1 WORKING |  |
|  | 5.2 OUTPUT |  |
| **6.** | **REFERENCES** | **60-61** |

**List of figures**

|  |  |
| --- | --- |
| Figure 3.1 | Non-Functional Requirements |
| Figure 3.2 | ALML DL |
| Figure 3.3 | Artificial Neural Network |
| Figure 3.4 | Software Development Life Cycle |
| Figure 3.5 | Steps of methodology |
| Figure 3.6 | Layers of Convolutional Neural Network |
| Figure 3.7 | Convolutional Matrix and Formulae |
| Figure 3.8 | CNN Sequence Classification |
| Figure 3.9 | Example of a Trained Network |
| Figure 4.1 | System Architecture |
| Figure 4.2 | Data Flow Diagram Level 0 |
| Figure 4.3 | Data Flow Diagram Level 1 |
| Figure 4.4 | Data Flow Diagram Level 2 |
| Figure 4.5 | Flow Chart |
| Figure 4.6 | Sequence Diagram |
| Figure 4.7 | Use Case Diagram |
| Figure 5.1 | Input |
| Figure 5.2 | Images after applying Gabor filter |
| Figure 5.3 | Sequential steps for data Modelling |
| Figure 5.4 | Training phase |
| Figure 5.5  Figure 5.6  Figure 5.7  Figure 5.8  Figure 5.9 | Accuracy epoch  Loss epoch  Confusion Matrix  Testing phase  The Final Result |
|  |  |

**CHAPTER-1**

**INTRODUCTION**

Alzheimer’s Disease (AD) is the most common cause of Dementia in people of the age 65 years and above. It is a progressive and irreversible neurological disease which follows a distinct pattern of brain damage as the disease progresses. The damage to the brain begins a decade before memory and other cognitive problems become evident. The plaques and Tangles are some of the main features in AD. The first region affected in the brain is the Hippocampus, which is responsible in forming memories and serving as a relay structure between the brain and body. In an Alzheimer’s affected brain, the hippocampus shrinks abnormally between 2.2 to 5.9 percent annually. The shrinkage of the hippocampus is mainly due to cell loss and damage of synapses. The destruction of synapses results in loss of ability of neurons to communicate with each other via signalling. Hence, in severe stages complications in. episodic and short term memory as well as dissolution of neural communications is observed.

The Alzheimer’s disease progresses slowly in three general stages: mild, moderate and severe. Since Alzheimer’s affects people in different ways, each person may experience symptoms or progression through the stages differently. In the early stage of Alzheimer’s, a person may function independently. Regardless to this, the person might feel that he or she is

Having memory lapses, for example forgetting familiar words or location of everyday objects. Middle-stage Alzheimer’s is typically the longest stage and can last for many years. During this stage, damage to nerve cells in the brain can lead to difficulty in expressing thoughts and performing routine tasks. The symptoms are also noticeable to others during this stage. In the last stage of the disease, the patient loses ability to respond to its environment, for carrying out conversations and, eventually, even to control movements. As memory and cognitive skills continue to worsen, individuals need extensive help with daily activities. Unfortunately, very few AD patients are diagnosed at an early stage. Imaging modalities such as Magnetic Resonance Imaging (MRI) scan, Positron Emission Tomography (PET) scan and Single-photon Emission Computed Tomography (SPECT) scan are used to track changes in the brain and diagnose Alzheimer’s before irreversible neural damage is done. The current no n-automated methods for diagnosis include cognitive impairment testing, mini-mental state examination (MMSE) and Clinical Dementia Rating (CDR). to others during this stage. In the last stage of the disease, the patient loses ability to respond to its environment, for carrying out conversations and, eventually, even to control movements. As memory and cognitive skills continue to worsen, individuals need extensive help with daily activities. Unfortunately, very few AD patients are diagnosed at an early stage. Imaging modalities such as Magnetic Resonance Imaging (MRI) scan, Positron Emission Tomography (PET) scan and Single-photon Emission Computed Tomography (SPECT) scan are used to track changes in the brain and diagnose Alzheimer’s before irreversible neural damage is done. The current no automated methods for diagnosis include cognitive impairment testing, mini-mental state examination (MMSE) and Clinical Dementia Rating (CDR).

In the proposed approach, various shape and texture features are extracted from the hippocampus for detection of Alzheimer’s disease. The texture features are extracted using Gray-Level Co-Occurrence Matrix (GLCM). GLCM is a method used for extracting second order statistical texture features. The shape features are extracted using the using the seven moment invariants. The Moment invariants define set properties of region that can be used for identification of shape. Based on the features extracted from the hippocampus, AD can be classified into various stages using a trained classifier. A neural network will be trained using Error Back Propagation (EBP) algorithm for the same.

The paper is organized as follows. Section 2 describes work related to detection of AD using MRI scans. The proposed approach is explained in section 3. Finally, section 4 concludes the proposed approach.

**CHAPTER-2**

**LITERATURE SURVEY**

**(1) Early Diagnosis of Alzheimer’s Disease Based on Resting-State Brain Networks and Deep Learning *-Ronghui Ju, Chenhui Hu and Pan Zhou***

This paper has proposed a framework for the early diagnosis of AD based on deep neural networks. The dataset has been acquired from ADNI (Alzheimer’s disease neuroimaging association database. Functional brain networks have been built based on the correlation of R-fMRI signals.

In this paper, they divided the brain into 90 regions and transform the R-fMRI data to a 90 X 130 matrix which retains the primary information. Each row describes the characteristics of different brain regions varying over time, and we refer to this matrix as the R-fMRI time series data. Next, in order to explore the functional connectivity between different brain regions, they used Pearson’s correlation coefficient to measure the strength of the links. Based on the correlation coefficient, the time series data is transformed into a 90 X 90 correlation coefficient matrix and a complete functional brain network is constructed. This network structure is sensitive to brain diseases since it represents the functional connectivity of the brain. The correlation coefficient data is our basis in detecting MCI.

In addition, the clinical examination data (including age, gender and genetic information) helps to analyse the relationship between MCI and other physiological factors. Then, a deep autoencoder network model is built to categorize these correlation coefficient data. Deep learning model based on stacked autoencoders have been developed to extract hierarchical features in high-dimensional data. Consequently, the deep autoencoder network effectively extracts the discriminative features from brain networks, and yields a high classification accuracy. This method greatly improves the capability of distinguishing MCIs and NCs as compared with the traditional approaches, and provides clinic decision support for neurodegenerative diseases, especially AD.

They used materials such as Images from the Alzheimer’s disease neuroimaging association (ADNI) as data base. They first select all the fMRI data in the ADNI-2 phase, since it contains most of the fMRI data, then they filter the data-based quality of image.

The methods used in this paper are R-fMRI classification model, construction of Brain Networks, Pre-training of Autoencoder such as initialising and optimising the parameters and fine tuning of Autoencoder.

The experimental results demonstrate the following models to extract discriminative brain features and classify the subjects: LDA, LR, SVM and the autoencoder network and the distribution pattern of patient's differs from gender and age.

**(2) Alzheimer Disease Detection and Tracking of Alzheimer Patient**

**- *Priyanka Thakare and Dr. V. RPawar***

In this paper they proposed to diagnosis of AD patient from EEG recording. In traditional day, diagnosis of AD can be possible using MMSE, CDR, blood test, spinal fluid, neurological examination. Analysis of EEG of MCI and AD patients can be possible on the basis of three factors i.e. slowing of EEG, reduced complexity of EEG signal perturbations in EEG synchrony and the relations between EEG measures & performance on test of global cognition, languages & executive functioning.

The variation in EEG recording in different stages can detect the AD patient and also, they proposed, some changes is occurring in EEG signal of various frequency bands of EEG signal in MCI patients & AD patients and also in the monitoring system, Alzheimer patient is track by using GPS and GSM. With the help of this monitoring system Alzheimer patient is travel anywhere without caregiver.

The methods used in this paper are

(1) Diagnosis of AD patient which includes the Participant, EEG database, Pre-processing in which the unwanted EEG signal is removed with sampling frequency of 50 Hz, Feature extraction and selection and finally the classification of AD and the normal person is based on the results.

(2) Tracking system for AD Patient in which the AD patient module has ARM cortex M3 LPC 1768, GPS antenna, power supply, 16\*2 LCD display, and GSM and GPS module. The caregiver module has GSM module, PC and RS 232 kit.

The result of the tracking system is showing the latitude and longitudinal co-ordinates and exact location of AD patient. From tracking system, caregiver is received the position of AD patient using GSM and GSP. In EEG signal several abnormities are measured such as slowing in EEG signal, complexity of EEG signal is decreases.

**3. Early Detection of Alzheimer's using digital image processing through Iridology, an Alternative method - *Fernando Hernández; Roberto Vega***

This paper talks about the Early detection of Alzheimer’s disease by the help of Iridology.

Over the years, alternative medicine has carried out numerous systematic studies of the eye, proposing as an analytical support to the iridology, which studies the alterations of the iris in correspondence with the organs of the human body. This is why it presents the opportunity to investigate and learn about certain neuronal diseases, specifically Alzheimer's based on digital image processing.

For this, the use of mathematical models based on specialized software (MATLAB), determines the characteristics of the iris establishing criteria or patterns to determine the existence or not of Alzheimer's.

For this, they have: i) The Fourier Transform to normalize the image;

ii) The Hough Transform to locate circles in an image.

In addition, some learning methods based on three multi-layer classifiers were used:

Zero R, Naïve Bayes and Multi-layer Perceptron.

The differentiation between the subtypes of pathologies should be investigated using well characterized and larger samples, that is to say that the image data should be more robust and with better resolution to avoid influencing the results. Since the potential of the image analysis and the quality of the data depend on the adjustment of the criteria and the decision making when issuing the diagnosis.

With the experimentation of digital processing algorithms, an infinite number of parameters can be determined that can be useful for any type of research. In this case, colour characteristics were taken, spectra for the determination of the characteristics necessary for compliance with this research.

During the process the classifier that obtains better results is Naïve Bayes with a 61.96% of correct diagnoses, a 74.00% chance that a sick patient obtains a diagnosis with a positive result, and a 47.62% that a healthy person is recognized of not having the evaluated medical condition. The error values allow making a decision about the model and permissible error range and adapting it according to the approximation criteria.

The data generated by the predictive classifier Naïve Bayes allows feedback to the generation process of the function that is being modelled.

In this paper, they were able to conclude that the Naïve Bayes classification generated the result with the maximum accuracy of 74% that a sick patient obtains a diagnosis with a positive result and a 47.62% that a healthy person is recognized of not having the evaluated medical condition.

**4. A Segmentation Technique to Detect the Alzheimer's Disease Using Image Processing - R. Anithat, Mr. Prakash, S. Jyothe**

This paper talks about the segmentation technique used to detect the Alzheimer’s disease using Image Processing techniques instead of Manual Segmentation.

An important area under medica research is Brain image analysis, results to detect brain diseases. The main causes for Alzheimer's diseases are Low brain activity and blood flow. In general Segmentation technique is using for the medical images.

One of the important components of the brain is Hippocampus. The normal behaviour of human beings is dependent on the functionality of Hippocampus. Manual Segmentation by a specialist on the Hippocampus takes many hours. In image processing there are various techniques available for segmentation process.

In this paper, a modified approach based on the watershed algorithm is used for segmenting the hippocampus region. The brain images converted into binary form using two approaches. The first approach is block mean, mask and labelling concepts and in the second approach top hat, mask and labelling concepts. However, it is found that some part of the image contains holes which interrupt the segmentation process.

To overcome this problem image hole filling techniques are implemented and related components are grouped into connected components. The shape analysis of hippocampus structure will result in classifying the Alzheimer's disease.

Watershed algorithm has been used for the segmentation of the MRI images. After the segmentation is carried out, resulting segments features could be extracted and subsequently classified. Classification could be carried out based on features such as grey, white, and cerebrospinal fluid (CSF) anatomically regions in brain.

An efficient watershed algorithm is preceded by using a marker image. A marker image defines the included zero marker values of watershed line pixels. For efficient Watershed segmentation a marker image needs to be accurately calculated. The markers are classified into two, internal and x external markers. Internal markers are imposed inside the objects to be identified; external markers are imposed outside the objects. Markers can be composed by various methods such as linear filtering, nonlinear filtering, or morphological processing. The choice usually is determined by the nature of the processed image.

Watershed algorithms are extremely vulnerable to noise. Watershed should present correct contours and may show other erroneous contours due to noise, therefore it may produce an over segmentation of the image.

In this paper, they were able to conclude that the final outcome of this watershed algorithm of the brain scan is analysed and the diseased area is analysed using the Shape analysis techniques. The classification technique can also apply for further classification of the image.

**5.Diagnosis of Alzheimer’s Disease using Machine Learning**

**-Priyanka Lodha, Ajay Talele and KishoriDegaonkar**

This Paper talks about a comprehensive algorithm that has been implemented to predict Alzheimer’s from available data like whole brain volume which is extracted from MRI scans and other cognitive and biological features. Clustering algorithms have also been implemented along with Fuzzy interference system. The algorithms Logistic Regression, Support Vector Machine, Gradient boosting, Neural Network, K-Nearest Neighbour, Random Forest are implemented.

Deterioration is defined as “the scientific problem of the cognitive energy is symbolized by the serious global decrease in mental work is especially not because of adjustment in the carefulness.

Data utilized in this paper is taken by Alzheimer’s Disease Neuroimaging Initiative (ADNI) index (http://adni.loni.usc.edu). Total details of information is taken from http://www.loni.usc.edu/ADNI/. “Very” premature stage of an Alzheimer disease. Diagnosis of the distinct causes an Alzheimer disease could be of great aid to the clinicians and researcher for development of new and effective treatments and monitoring mechanisms. The ages between 55 and 90 were applied in the ADNI study.

In order to detect Alzheimer’s Subjects and analyse images Alzheimer Disease related regions of brain, we use MRI images and process them to get numeric data which in turn is processed using machine learning algorithms. Neural Network and Random Forest have much better performs in accuracy, then other used methods.

The implementation of this method will give instant and accurate results. Support Vector Machine and Gradient boosting are also powerful algorithms for classification problems and it works well with the problem at hand. At a premature stage Alzheimer disease can be detected and necessary treatment done at these early states it will minimize the possibility of creating further complications of Alzheimer disease patients.

Machine Learning is an application of data analysis. These algorithms try to build a model that predicts desired output for available data through statistical and predictive features. These models can identify patterns and correlations in data provided and eventually try to make decisions without or with minimal intervention from human. We determine to design a Machine Learning model to accurately predict Alzheimer’s disease of a person from given parameters which consist of different cognitive and medical factors.

In this paper we were able to conclude that the CNN algorithm will provide better results for the classification of Alzheimer’s Disease.

**6. FEATURES EXTRACTION USING A GABOR FILTER FAMILY**

**Danian Zheng and Yannan Zhao**

Gabor filters possess the optimal localization properties in both spatial and frequency domain, and they have been successfully used in many applications. But how to design a set of befitting Gabor filters for a specific application has maybe puzzled many users for a long time. This paper purpose designing a common set of Gabor filters – a Gabor filter family to solve the problem.

The Gabor filters in the family are well combined to capture the whole frequency spectrum in all directions and extracting many meaningful features using the Gabor filter family. Experimental results in textures and characters demonstrate these features commendably expressing the local information with the different frequencies and orientations in the image. The Gabor filter family designed by us can be also used in some other applications.

Designing some befitting Gabor filters for a particular processing task and reducing the computation time are the common difficulties, and they should be well solved in all these applications. In texture processing, some authors consider the design of a single Gabor filter to segment a two-texture image. The output of a Gabor-filtered texture is modelled well by a Rician distribution, and a measure of total output power is used to select the centre frequency of filter and to estimate the Rician statistics of the Gabor- filtered image.

The features extracted using Gabor filters represent the local information in the image. We can take features from the amplitude or phase after convolving the complex Gabor filters with the image. In this paper, we give out four feature examples. In order to reduce the computational complexity, we can choose the numbers of Gabor filters (frequency number and orientation number), features and convolving coefficients (decimation).

In recent years, Gabor filters have been successfully used in many applications, such as texture segmentation/classification ,target detection, character recognition ,fingerprint recognition , face recognition , fractal dimension management, document analysis, edge detection , image analysis and compression .Gabor filters have received considerable attention, because they possess the optimal localization properties in both spatial and frequency domain.

A Gabor filter family is very similar to a wavelet family, and the most difference between them is the mode of decomposing the 2-D frequency domain: the former is circle mode, but the latter is square mode. And thus, Gabor filters have the orientation characteristics. We have designed three Gabor filter families and discussed the differences in them, and we commend the third one to users.

In this Paper they were able to come to a conclusion that for Edge detection Gabor filter Perform better with respect to the other filters and It will increase the accuracy of the model.

**CHAPTER-3**

**METHODOLOGY**

**Proposed System**

In the proposed approach, various shape and texture features are extracted from the hippocampus for detection of Alzheimer’s disease. From the training dataset features are extracted using Convolution Neural Network (CNN) an advanced deep learning method and a training model is processed and it gives a validation model.

Test data has to give to validation model which was build using CNN, the validation model has to extract the features from the input MRI scan image and classify the Alzheimer category.

**Advantages of Proposed System:**

**CNN is a new upcoming technology**

**Accuracy is high**

## **Functional Requirements:**

A function of software system is defined in functional requirement and the behaviour of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality. The functional requirements of the project are one of the most important aspects in terms of entire mechanism of modules.

## **Non-Functional Requirements:**

Non-functional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system’s *quality attributes*. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

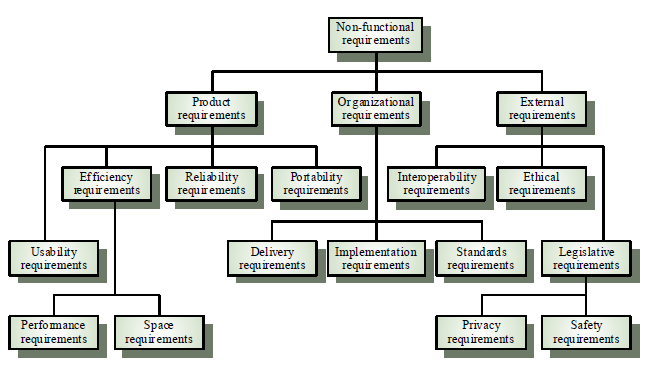
****

Fig 2.1- Non-Functional Requirements

**Some Non-Functional Requirements are as follows:**

**Reliability**

The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be Project pioneer and in addition the Test designer.

**Maintainability**

The system watching and upkeep should be fundamental and focus in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.

**Performance**

The framework will be utilized by numerous representatives all the while. Since the system will be encouraged on a single web server with a lone database server outside of anyone's ability to see, execution transforms into a significant concern. The structure should not capitulate when various customers would use everything the while. It should allow brisk accessibility to each and every piece of its customers. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not to be any irregularity at the same time.

**Portability**

The framework should to be effectively versatile to another framework. This is obliged when the web server, which s facilitating the framework gets adhered because of a few issues, which requires the framework to be taken to another framework.

**Scalability**

The framework should be sufficiently adaptable to include new functionalities at a later stage. There should be a run of the mill channel, which can oblige the new functionalities.

**Flexibility**

Flexibility is the capacity of a framework to adjust to changing situations and circumstances, and to adapt to changes to business approaches and rules. An adaptable framework is one that is anything but difficult to reconfigure or adjust because of diverse client and framework prerequisites. The deliberate division of concerns between the trough and motor parts helps adaptability as just a little bit of the framework is influenced when strategies or principles change

**Hardware Requirements:**

**System** : Pentium IV 2.4 GHz.

**Hard Disk**  : 500 GB.

**Ram**  : 4 GB

*Any desktop / Laptop system with above configuration or higher level*

**Software Requirements:**

**Operating system** : Windows 7 / 8 / 10

**Coding Language** : Python

**Software** : Anaconda

**IDE** :Jupyter Notebook

**Python Introduction:**

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

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

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

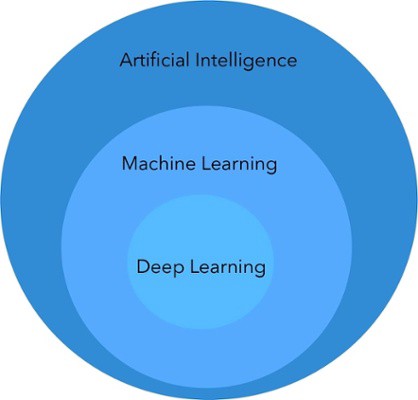
**Python is Interactive −** you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

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

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

**Artificial intelligence**

Artificial Intelligence encompasses a very broad scope. You could even consider something like Dijkstra's shortest path algorithm as Artificial Intelligence. However, two categories of AI are frequently mixed up: Machine Learning and Deep Learning. Both of these refer to statistical modelling of data to extract useful information or make predictions. In this article, we will list the reasons why these two statistical modelling techniques are not the same and help you further frame your understanding of these data modelling paradigms.

Fig 2.2- AI,ML,DL

**Machine learning**

Machine Learning is a method of statistical learning where each instance in a dataset is described by a set of features or attributes. In contrast, the term “Deep Learning” is a method of statistical learning that extracts feature or attributes from raw data. Deep Learning does this by utilizing neural networks with many hidden layers, big data, and powerful computational resources. The terms seem somewhat interchangeable, however, with Deep Learning method, The algorithm constructs representations of the data automatically. In contrast, data representations are hard-coded as a set of features in machine learning algorithms, requiring further processes such as feature selection and extraction, (such as PCA).

Both of these terms are in dramatic contrast with another class of classical artificial intelligence algorithms known as Rule-Based Systems where each decision is manually programmed in such a way that it resembles a statistical model.

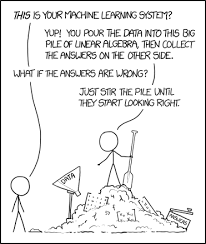
In Machine Learning and Deep Learning, there are many different models that fall into two different categories, supervised and unsupervised. In unsupervised learning, algorithms such as k-Means, hierarchical clustering, and Gaussian mixture models attempt to learn meaningful structures in the data. Supervised learning involves an output label associated with each instance in the dataset. This output can be discrete/categorical or real-valued. Regression models estimate real-valued outputs, whereas classification models estimate discrete-valued outputs. Simple binary classification models have just two output labels, 1 (positive) and 0 (negative). Some popular supervised learning algorithms that are considered Machine Learning: are linear regression, logistic regression, decision trees, support vector machines, and neural networks, as well as non-parametric models such as k-Nearest Neighbour’s.

**Data Size:**

Both Machine Learning and Deep Learning are able to handle massive dataset sizes, however, machine learning methods make much more sense with small datasets. For example, if you only have 100 data points, decision trees, k-nearest neighbour’s, and other machine learning models will be much more valuable to you than fitting a deep neural network on the data. This is due to the next topic of difference, Interpretability.

**Interpretability:**

**Example for how interpretability works in ML & DL:**



A lot of the criticism of deep learning methods and machine learning algorithms such as Support Vector Machine or (maybe, because you can at least visualize the constituent probabilities making up the output), Naive Bayes, are due to their difficulty to interpret. For example, when a Convolutional Neural Network outputs ‘cat’ in a dog vs. cat problem, nobody seems to know why it did that. In contrast, when you are modelling data such as an electronic health record or bank loan dataset with a machine learning technique, it is much easier to understand the reasoning for the model’s prediction.

One of the best examples of interpretability is decision trees where you follow logical tests down nodes of the tree until you reach a decision. Another machine learning algorithm with high interpretability is k-Nearest Neighbour’s. This is not a parametric learning algorithm but still falls under the category of machine learning algorithms. It is very interpretability because you easily reason about the similar instances for yourself.

**ARTIFICIAL NEURAL NETWORK:**

Artificial neural networks are one of the main tools used in machine learning. As the “neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize. While neural networks (also called “perceptron’s”) have been around since the 1940s, it is only in the last several decades where they have become a major part of artificial intelligence. This is due to the arrival of a technique called “back propagation,” which allows networks to adjust their hidden layers of neurons in situations where the outcome doesn’t match what the creator is hoping for — like a network designed to recognize dogs, which misidentifies a cat, for example.

Another important advance has been the arrival of deep learning neural networks, in which different layers of a multilayer network extract different features until it can recognize what it is looking for.

**Historical background of NN:**

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

Many import and advances have been boosted by the use of inexpensive computer emulations. Following an initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important advances were made by relatively few researchers. These pioneers were able to develop convincing technology which surpassed the limitations identified by Minsky and Paper, published a book (in 1969) in which they summed up a general feeling of frustration (against neural networks) among researchers, and was thus accepted by most without further analysis. Currently, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

**Why use neural networks?**

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

* **Adaptive learning:**

An ability to learn how to do tasks based on the data given for training or initial experience. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.

* **Real Time Operation:**

ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

**Fault Tolerance via Redundant Information Coding:**

Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

**Neural networks versus conventional computers:**

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem-solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high-level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

**Neuron perceptron:**

The perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector.

**Multilayer perceptron:**

A multilayer perceptron (MLP) is a class of feed forward artificial neural network. A MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.

The MLP consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinearly-activating nodes. Since MLPs are fully connected, each node in one layer connects with a certain weight to every node.

**How neural networks figured and described in deep learning project:**

Another important advance has been the arrival of deep learning neural networks, in which different layers of a multilayer network extract different features until it can recognize what it is looking for.

The next layer could then identify any edges in the image, based on lines of similar pixels. After this, another layer may recognize textures and shapes, and so on. By the time the fourth or fifth layer is reached, the deep learning net will have created complex feature detectors. It can figure out that certain image elements (such as a pair of eyes, a nose, and a mouth) are commonly found together.

Once this is done, the researchers who have trained the network can give labels to the output, and then use back propagation to correct any mistakes which have been made. After a while, the network can carry out its own classification tasks without needing humans to help every time.

Beyond this, there are different types of learning, such as supervised or unsupervised learning or reinforcement learning, in which the network learns for itself by trying to maximize its score — as memorably carried out by Google Deep Mind’s Atari game-playing bot.

**Basic structure of ANN:**

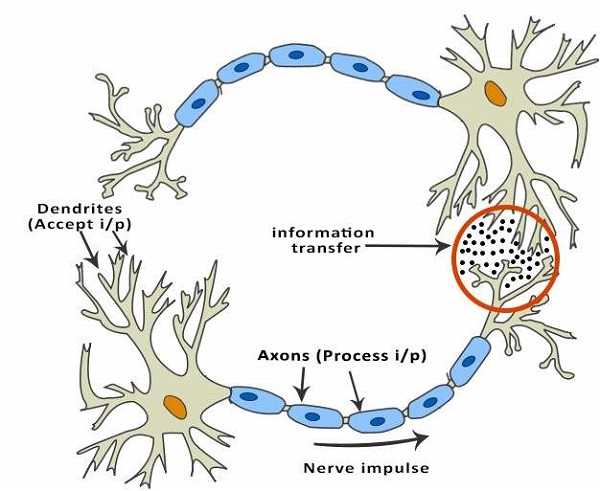


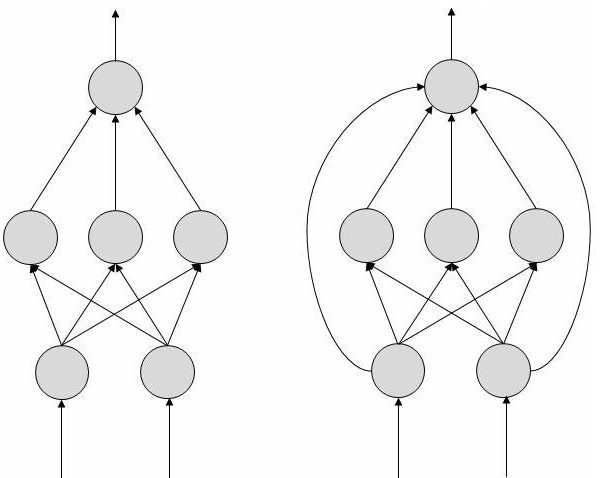
Fig-3.3- Artificial Neural Network

**Types of Artificial Neural Networks:**

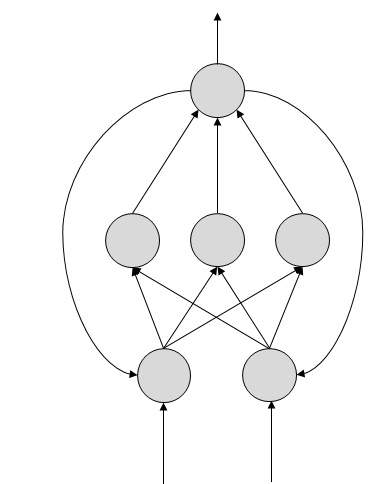
There are two Artificial Neural Network topologies – Feed Forward and Feedback.

**Feed Forward ANN:**

In this ANN, the information flow is unidirectional. A unit sends information to other unit from which it does not receive any information. There are no feedback loops. They are used in pattern generation/recognition/classification. They have fixed inputs and outputs.



**Feed Back ANN:** A feedback-based prediction is defined as approximating an ultimate outcome in an iterative manner where each iteration’s operation is based upon the present outcome.



**Applications of Neural Networks:**

**They can perform tasks that are easy for a human but difficult for a machine**

Aerospace − Autopilot aircrafts, aircraft fault detection.

Automotive − Automobile guidance systems.

Military − Weapon orientation and steering, target tracking, object discrimination, facial recognition, signal/image identification.

Electronics − Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.

Financial − Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document readers, credit application evaluators.

Industrial − Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modelling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.

Medical − Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.

Speech − Speech recognition, speech classification, text to speech conversion.

Telecommunications − Image and data compression, automated information services, real-time spoken language translation.

Transportation − Truck Brake system diagnosis, vehicle scheduling, routing systems.

Software − Pattern Recognition in facial recognition, optical character recognition, etc.

Time Series Prediction − ANNs are used to make predictions on stocks and natural calamities.

Signal Processing − Neural networks can be trained to process an audio signal and filter it appropriately in the hearing aids.

Control − ANNs are often used to make steering decisions of physical vehicles.

**Django web framework:**

**Introduction:**

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

**Why Django?**

With Django, you can take Web applications from concept to launch in a matter of hours. 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’s free and open source.

Advantages of Django framework for python project:

Ridiculously fast: Django was designed to help developers take application from concept to completion as quickly as possible.

Fully loaded: Django includes dozens of extras you can use to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

Reassuringly secure: Django takes security seriously and helps developers avoid many common security mis takes, such as SQL injection, cross-site scripting, cross-site request forgery and click jacking. Its user authentication system provides a secure way to manage user accounts and passwords.

Exceedingly scalable: Some of the busiest sites on the planet use Django’s ability to quickly and flexibly scale to meet the heaviest traffic demands.

Incredibly versatile: Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

How Django framework works with python project:

Based on these:

The model layer or abstraction layer.

The view layer

The template layer

Forms

The development processes

The admin

Security

Internationalization and localization

Performance and optimization

Geographic framework

Common web application tools

Authentication: Overview | Using the authentication system | Password management | Customizing authentication | API Reference

Caching

Logging

Sending emails

Syndication feeds (RSS/Atom)

Pagination

Messages framework

Serialization

Sessions

Sitemaps

Static files management

Data validation

Other core functionalities

Conditional content processing

Content types and generic relations

Flat pages

Redirects

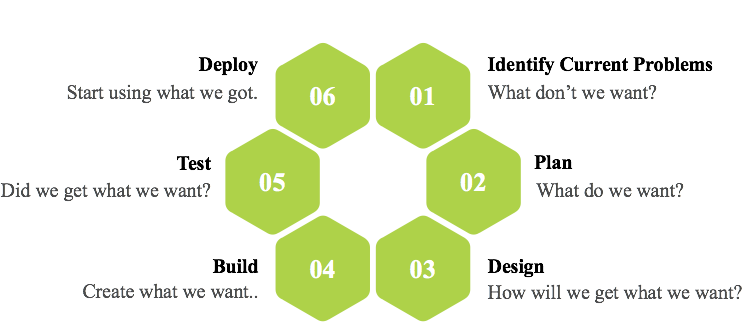
Signals

System check framework

The sites framework Unicode in Django

**SDLC**

It will cover the details explanation of methodology that is being used to make this project complete and working well. Many methodology or findings from this field mainly generated into journal for others to take advantages and improve as upcoming studies. The method is use to achieve the objective of the project that will accomplish a perfect result. In order to evaluate this project, the methodology based on System Development Life Cycle (SDLC), generally three major step, which is planning, implementing and analysis.

****

**Fig 3.4: Software Development Life Cycle**

****

**Fig 3.5: Steps of Methodology**

**Planning:**

To identify all the information and requirement such as hardware and software, planning must be done in the proper manner. The planning phase has two main elements namely ***data collection*** and ***the requirements of hardware and software***.

**Data collection:**

Machine learning needs two things to work, data (lots of it) and models. When acquiring the data, be sure to have enough features (aspect of data that can help for a prediction, like the surface of the house to predict its price) populated to train correctly your learning model. In general, the more data you have the better so make to come with enough rows.

The primary data collected from the online sources remains in the raw form of statements, digits and qualitative terms. The raw data contains error, omissions and inconsistencies. It requires corrections after careful scrutinizing the completed questionnaires. The following steps are involved in the processing of primary data. A huge volume of raw data collected through field survey needs to be grouped for similar details of individual responses.

Data Pre-processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

Therefore, certain steps are executed to convert the data into a small clean data set. This technique is performed before the execution of Iterative Analysis. The set of steps is known as Data Pre-processing. It includes -

Data Cleaning

Data Integration

Data Transformation

Data Reduction

Data Pre-processing is necessary because of the presence of unformatted real-world data. Mostly real-world data is composed of -

**Inaccurate data (missing data) -** There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics and much more.

**The presence of noisy data (erroneous data and outliers) -** The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and much more.

**Inconsistent data -** The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more.

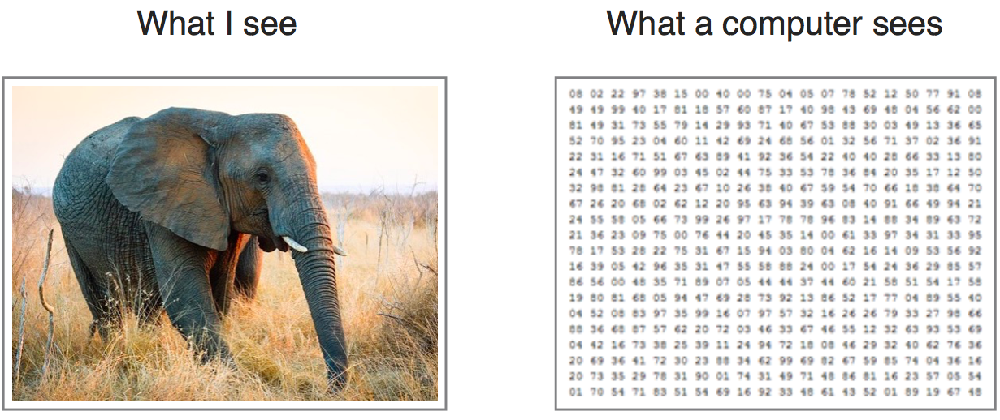
**Implementing:**

In this work, a business intelligent model has been developed, to classify different animals, based on a specific business structure deal with Animal classification using a suitable machine learning technique. The model was evaluated by a scientific approach to measure accuracy. We are using Convolutional Neural Network (CNN) to build our model.

**Convolutional Neural Network**

A convolutional neural network (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example, Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google for search through among users’ photos.

Let us consider the use of CNN for image classification in more detail. The main task of image classification is acceptance of the input image and the following definition of its class. This is a skill that people learn from their birth and are able to easily determine that the image in the picture is an elephant. But the computer sees the pictures quite differently:

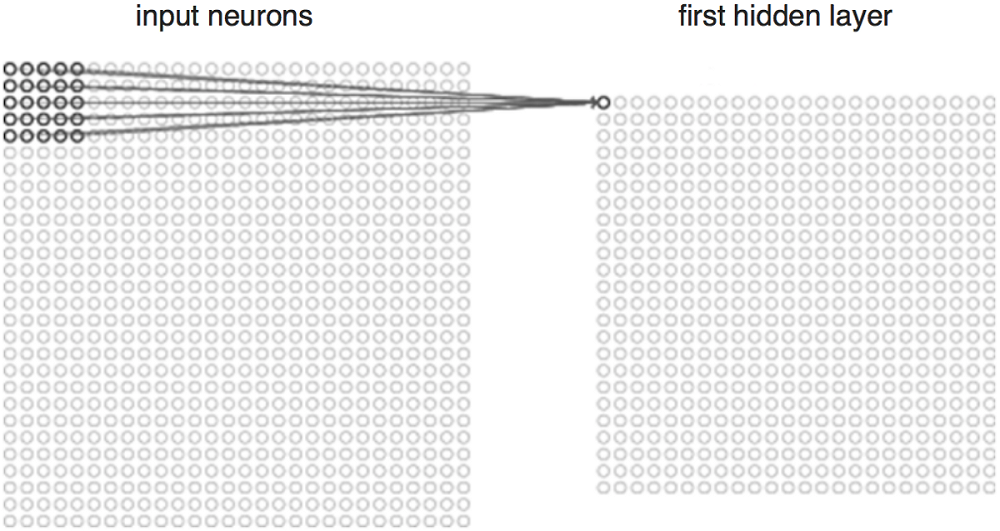


Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.

To solve this problem the computer looks for the characteristics of the base level. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.

**In more detail:** the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

**The** **Convolution layer** is always the first. The image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of image. Next the software selects a smaller matrix there, which is called a **filter**(or neuron, or core). Then the filter produces convolution, i.e. moves along the input image. The filter’s task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller then an input matrix.



This operation, from a human perspective, is analogous to identifying boundaries and simple colours on the image. But in order to recognize the properties of a higher level such as the trunk or large ears the whole network is needed.

**The network** will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.

**The nonlinear layer** is added after each convolution operation. It has an activation function, which brings nonlinear property. Without this property a network would not be sufficiently intense and will not be able to model the response variable (as a class label).

**The pooling layer** follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them. As a result, the image volume is reduced. This means that if some features (as for example boundaries) have already been identified in the previous convolution operation, then a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

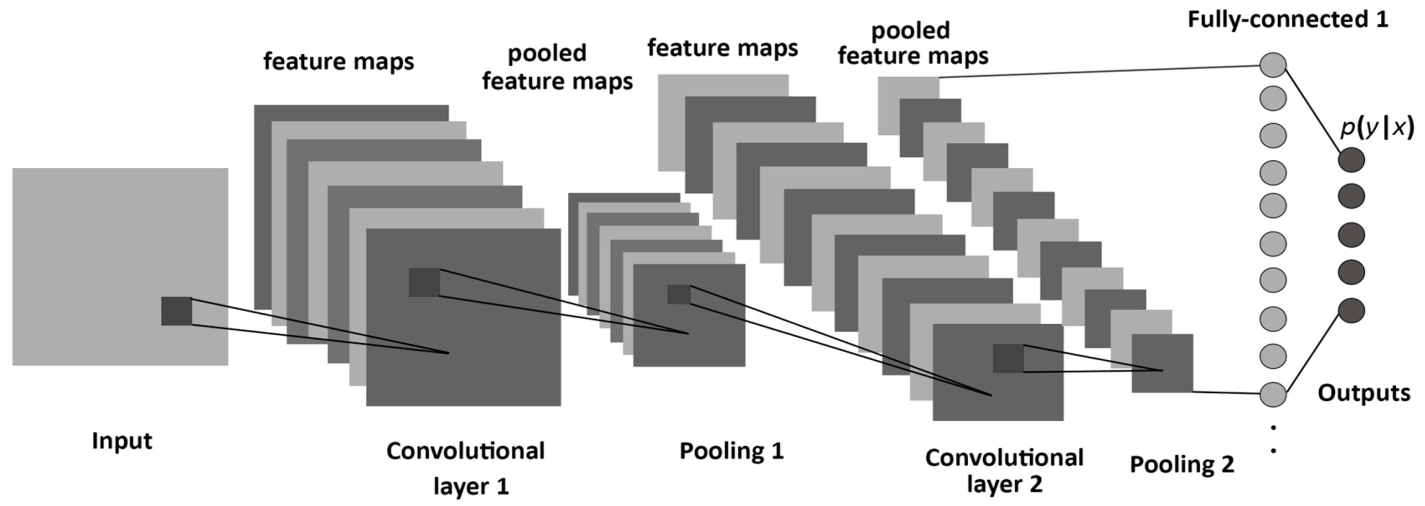


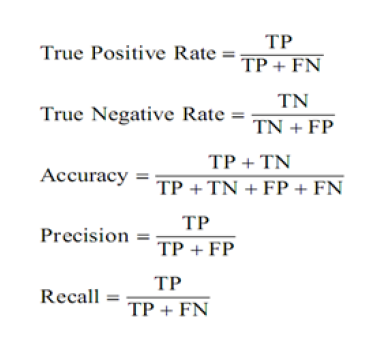
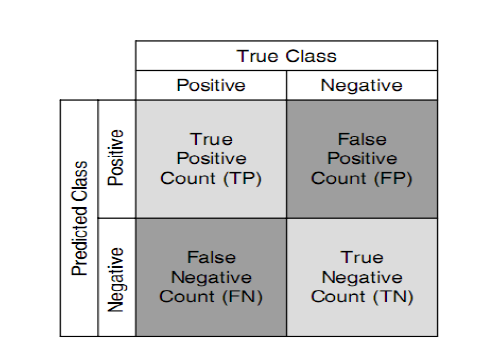
Fig 3.6- Layers of CNN

After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach a fully connected layer. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the number of classes from which the model selects the desired class.

**Analysis:**

In this final phase, we will test our classification model on our prepared image dataset and also measure the performance on our dataset. To evaluate the performance of our created classification and make it comparable to current approaches, we use accuracy to measure the effectiveness of classifiers.

After model building, knowing the power of model prediction on a new instance, is very important issue. Once a predictive model is developed using the historical data, one would be curious as to how the model will perform on the data that it has not seen during the model building process. One might even try multiple model types for the same prediction problem, and then, would like to know which model is the one to use for the real-world decision making situation, simply by comparing them on their prediction performance (e.g., accuracy). To measure the performance of a predictor, there are commonly used performance metrics, such as accuracy, recall etc. First, the most commonly used performance metrics will be described, and then some famous estimation methodologies are explained and compared to each other. "Performance Metrics for Predictive Modelling In classification problems, the primary source of performance measurements is a coincidence matrix (**classification matrix or a contingency table**)”. Above figure shows a coincidence matrix for a two-class classification problem. The equations of the most commonly used metrics that can be calculated from the coincidence matrix are also given in Fig 2.7.



**Figure 3.7: confusion matrix and formulae**

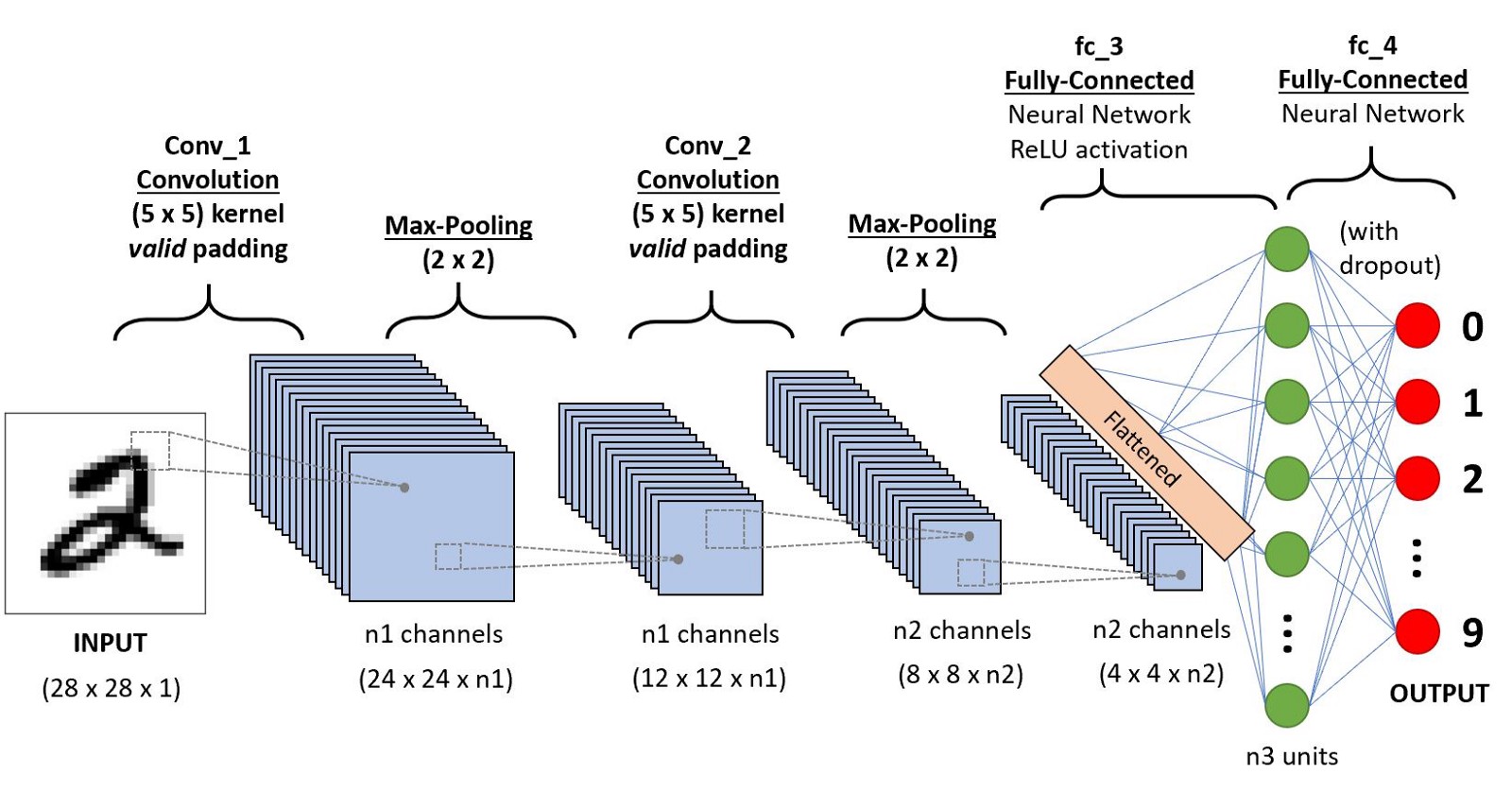
As being seen in above figure, the numbers along the diagonal from upper-left to lower-right represent the correct decisions made, and the numbers outside this diagonal represent the errors. "The true positive rate (also called hit rate or recall) of a classifier is estimated by dividing the correctly classified positives (the true positive count) by the total positive count. The false positive rate (also called a false alarm rate) of the classifier is estimated by dividing the incorrectly classified negatives (the false negative count) by the total negatives. The overall accuracy of a classifier is estimated by dividing the total correctly classified positives and negatives by the total number of samples.

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. A collection of such fields overlaps to cover the entire visual area.

CNNs have wide applications in image and video recognition, recommender systems and natural language processing. In this article, the example that I will take is related to Computer Vision. However, the basic concept remains the same and can be applied to any other use-case!

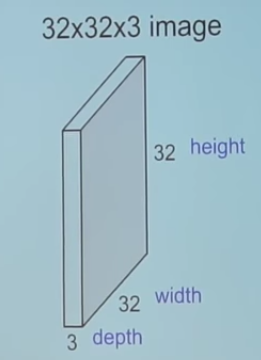
CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs. Pretty straightforward, right?

Fig 3.8- CNN sequence to classify handwritten digit:



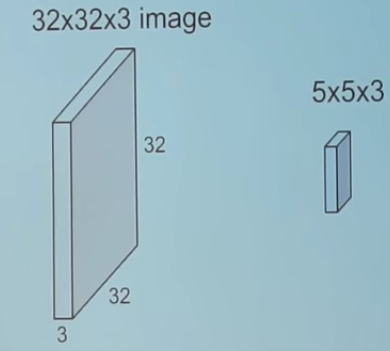
**So, how are Convolutional Neural Networks different than Neural Networks?**

**CNNs operate over Volumes:**

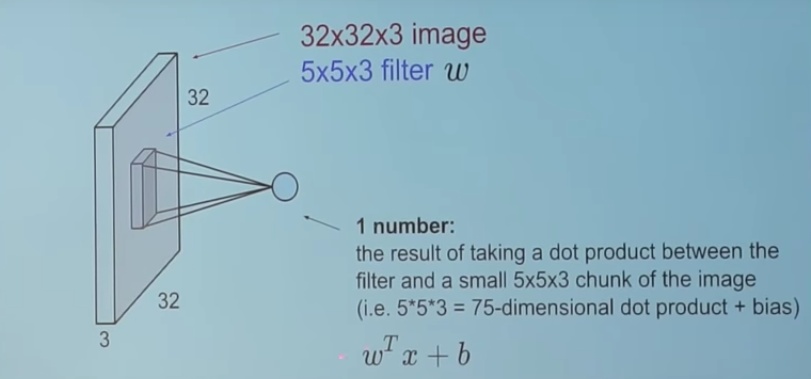


Unlike neural networks, where the input is a vector, here the input is a multi-channelled image (3 channelled in this case).

**What convolution means:**

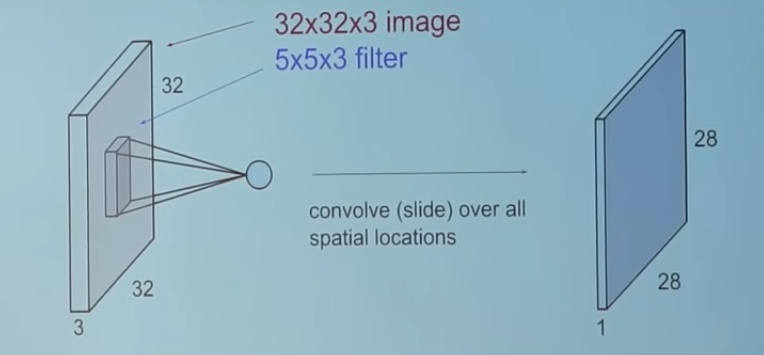


We take the 5\*5\*3 filter and slide it over the complete image and along the way take the dot product between the filter and chunks of the input image.

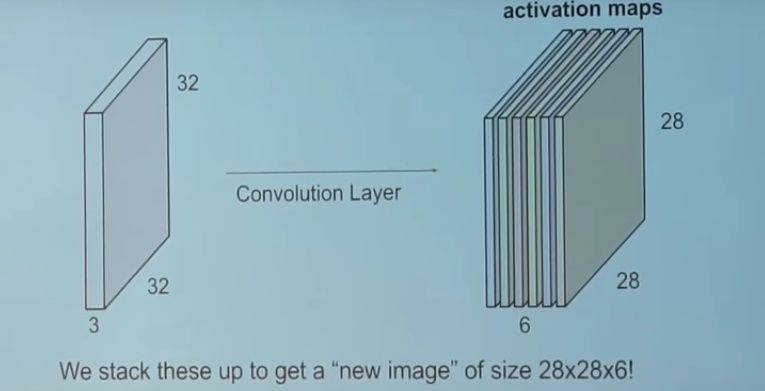


For every dot product taken, the result is a scalar.

So, what happens when we convolve the complete image with the filter?

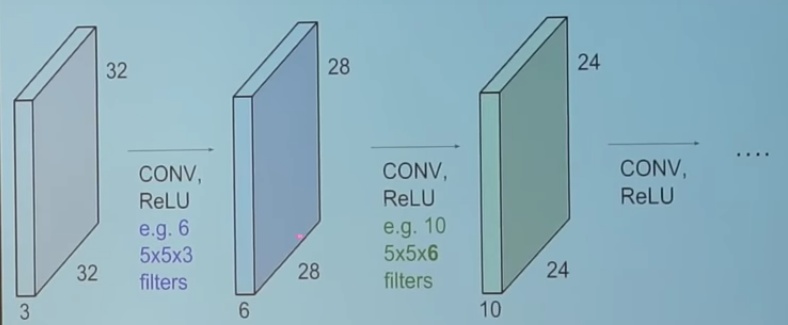


The convolution layer is the main building block of a convolutional neural network.



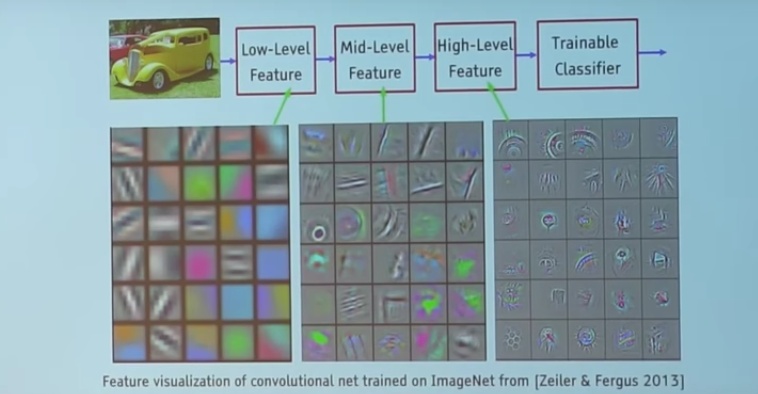
The convolution layer comprises of a set of independent filters (6 in the example shown). Each filter is independently convolved with the image and we end up with 6 feature maps of shape 28\*28\*1.

Suppose we have a number of convolution layers in sequence. What happens then?



All these filters are initialized randomly and become our parameters which will be learned by the network subsequently.

**Fig-3.9 Example of a trained network:**



Take a look at the filters in the very first layer (these are our 5\*5\*3 filters). Through back propagation, they have tuned themselves to become blobs of coloured pieces and edges. As we go deeper to other convolution layers, the filters are doing dot products to the input of the previous convolution layers. So, they are taking the smaller coloured pieces or edges and making larger pieces out of them.

Take a look at image 4 and imagine the 28\*28\*1 grid as a grid of 28\*28 neurons. For a particular feature map (the output received on convolving the image with a particular filter is called a feature map), each neuron is connected only to a small chunk of the input image and all the neurons have the same connection weights. So again, coming back to the differences between CNN and a neural network.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

LeNet

AlexNet

VGGNet

GoogLeNet

ResNet

ZFNet

**Keras**

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](https://github.com/tensorflow/tensorflow), [CNTK](https://github.com/Microsoft/cntk), or [Theano](https://github.com/Theano/Theano). It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).

Supports both convolution networks and recurrent networks, as well as combinations of the two.

Runs seamlessly on CPU and GPU.

Keras is compatible with: **Python 2.7-3.6**.

**Guiding principles**

**User friendliness.** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

**Modularity.** A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new models.

**Easy extensibility.** New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

**Work with Python**. No separate model’s configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

The focus of Keras is the idea of a model.

The main type of model is called a Sequence which is a linear stack of layers.

You create a sequence and add layers to it in the order that you wish for the computation to be performed.

Once defined, you compile the model which makes use of the underlying framework to optimize the computation to be performed by your model. In this you can specify the loss function and the optimizer to be used.

Once compiled, the model must be fit to data. This can be done one batch of data at a time or by firing off the entire model training regime. This is where all the compute happens.

Once trained, you can use your model to make predictions on new data.

We can summarize the construction of deep learning models in Keras as follows:

1. **Define your model**. Create a sequence and add layers.
2. **Compile your model**. Specify loss functions and optimizers.
3. **Fit your model**. Execute the model using data.
4. **Make predictions**. Use the model to generate predictions on new data.

**Flexibility**

Sometimes you just don’t want to use what is already there but you want to define something of your own (for example a cost function, a metric, a layer, etc.).   
Although Keras 2 has been designed in such a way that you can implement almost everything you want but we all know that low-level libraries provides more flexibility. Same is the case with TF. *You can tweak* TF much more as compared to Keras.

**Functionality**

Although Keras provides all the general-purpose functionalities for building Deep learning models, it doesn’t provide as much as TF. TensorFlow offers more advanced operations as compared to Keras. This comes very handy if *you are doing a research or developing some special kind of deep learning models. Some examples regarding high level operations are****:***

**Threading and Queues**

Queues are a powerful mechanism for computing tensors asynchronously in a graph. Similarly, you can execute multiple threads for the same Session for parallel computations and hence speed up your operations.

**Debugger**

Another extra power of TF. With TensorFlow, you get a specialized debugger. It provides visibility into the internal structure and states of running TensorFlow graphs. Insights from debugger can be used to facilitate debugging of various types of bugs during both training and inference.

**Control**

The more control you have over your network, more better understanding you have of what’s going on with your network.  
With TF, you get such a control over your network. You can control whatever you want in your network. Operations on weights or gradients can be done like a charm in TF.

**OPEN CV**

OpenCV was started at Intel in 1999 by Gary Bradskyand the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel’s Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, and Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

Python is a general-purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images.

**OpenCV Applications**

OpenCV is being used for a very wide range of applications which include:

Street view image stitching

Automated inspection and surveillance

Robot and driver-less car navigation and control

Medical image analysis

Video/image search and retrieval

Movies - 3D structure from motion

Interactive art installations

**OpenCV Functionality**

Image/video I/O, processing, display (*core, imgproc, highgui*)

Object/feature detection (*objdetect, features2d, nonfree*)

Geometry-based monocular or stereo computer vision (*calib3d, stitching, videostab*)

Computational photography (*photo, video, superres*)

Machine learning & clustering (*ml, flann*)

CUDA acceleration (*gpu*)

**Matplotlib**

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

**Installation:**  
Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package:

python -mpip install -U matplotlib

**Importing matplotlib:**

from matplotlib import pyplot as plt

*or*

import matplotlib.pyplot as plt

**Basic plots in Matplotlib:**

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They’re typically instruments for reasoning about quantitative information.

**Uses of matplotlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-**oriented** API for **embedding** plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Matplotlib is a Python package for 2D plotting that generates production-quality graphs. It supports interactive and non-interactive plotting, and can save images in several output formats (PNG, PS, and others). It can use multiple window toolkits (GTK+, wxWidgets, Qt, and so on) and it provides a wide variety of plot types (lines, bars, pie charts, histograms, and many more).

In addition to this, it is highly customizable, flexible, and easy to use. The dual nature of Matplotlib allows it to be used in both interactive and non-interactive scripts. It can be used in scripts without a graphical display, embedded in graphical applications, or on web pages. It can also be used interactively with the Python interpreter or IPython.

**Merits of Matplotlib**

The idea behind Matplotlib can be summed up in the following motto as quoted by John Hunter, the creator and project leader of Matplotlib:

“Matplotlib tries to make easy things easy and hard things possible”.

Matplotlib was born in the scientific area of computing, where gnu plot and MATLAB were (and still are) used a lot.

Matplotlib was modelled on MATLAB, because graphing was something that MATLAB did very well. The high degree of compatibility between them made many people move from MATLAB to Matplotlib, as they felt like home while working with Matplotlib.

But what are the points that built the success of Matplotlib? Let's look at some of them:

**It uses Python:** Python is a very interesting language for scientific purposes (it's interpreted, high-level, easy to learn, easily extensible, and has a powerful standard library) and is now used by major institutions such as NASA, JPL, Google, DreamWorks, Disney, and many more.

**It's open source, so no license to pay:** This makes it very appealing for professors and students, who often have a low budget.

**It's a real programming language:** The MATLAB language (while being Turing-complete) lacks many of the features of a general-purpose language like Python.

**It's much more complete:** Python has a lot of external modules that will help us perform all the functions we need to. So it's the perfect tool to acquire data, elaborate the data, and then plot the data.

**It's very customizable and extensible:** Matplotlib can fit every use case because it has a lot of graph types, features, and configuration options.

**It's integrated with LaTeX mark-up:** This is really useful when writing scientific papers.

**It's cross-platform and portable:** Matplotlib can run on Linux, Windows, Mac OS X, and Sun Solaris (and Python can run on almost every architecture available).

The aim of Matplotlib is to generate graphs. So, we need a way to actually view these images or even to save them to files. We're going to look at the various output formats available in Matplotlib and the graphical user interfaces (GUIs) supported by the library.

Matplotlib supports both the categories, particularly with the following output formats:

|  |  |  |
| --- | --- | --- |
| **Format** | **Type Description** | **Description** |
| EPS | Vector | Encapsulated PostScript |
| JPG | Raster | Graphic format with lossy compression method for photographic output. |
| PDF | Vector | Portable Document Format (PDF). |
| PNG | Raster | Portable Network Graphics (PNG), a raster graphics format with a lossless compression method (more adaptable to line art than JPG). |
| PS | Vector | Language widely used in publishing and as printers jobs format. |
| SVG | Vector | Scalable Vector Graphics (SVG), XML based. |

**BACKENDS**

A backend that displays the image on screen is called a user interface backend.

The backend is that part of Matplotlib that works behind the scenes and allows the software to target several different output formats and GUI libraries (for screen visualization).

In order to be even more flexible, Matplotlib introduces the following two layers structured (only for GUI output):

**The renderer:** This actually does the drawing

**The canvas:** This is the destination of the figure.

The standard renderer is the Anti-Grain Geometry (AGG) library, a high-performance rendering engine which is able to create images of publication level quality, with anti-aliasing, and sub pixel accuracy. AGG is responsible for the beautiful appearance of Matplotlib graphs.

The canvas is provided with the GUI libraries, and any of them can use the AGG rendering, along with the support for other rendering engines (for example, GTK+).

Let's have a look at the user interface toolkits and their available renderers:

|  |  |
| --- | --- |
| **Backend** | **Description** |
| GTKAgg | GTK+ (The GIMP ToolKit GUI library) canvas with AGG rendering. |
| GTK | GTK+ canvas with GDK rendering. GDK rendering is rather primitive, and doesn't include anti-aliasing for the smoothing of lines. |
| GTKCairo | GTK+ canvas with Cairo rendering. |
| WxAgg | wxWidgets (cross-platform GUI and tools library for GTK+, Windows, and Mac OS X. It uses native widgets for each operating system, so applications will have the look and feel that users expect on that operating system) canvas with AGG rendering. |
| WX | wxWidgets canvas with native wxWidgets rendering. |
| TkAgg | TkAgg Tk (graphical user interface for Tcl and many other dynamic languages) canvas with AGG rendering. |

**S K LEARN**

**Introduction**

scikit-learn is a library, i.e. a collection of classes and functions that users import into Python programs. Using scikit-learn therefore requires basic Python programming knowledge. No command-line interface, let alone a graphical user interface, is offered for non-programmer users

Scikit-learn is the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Scikit-learn is an open source Python library that has powerful tools for data analysis and data mining. It's available under the BSD license and is built on the following machine learning libraries: NumPy, a library for manipulating multi-dimensional arrays and matrices.

**Core API**

All objects within scikit-learn share a uniform common basic API consisting of three complementary interfaces: an estimator interface for building and fitting models, a **predictor** interface for making predictions and a transformer interface for converting data. In this section, we describe these three interfaces, after reviewing our general principles and data representation choices.

General principles As much as possible, our design choices have been guided so as to avoid the proliferation of framework code. We try to adopt simple conventions and to limit to a minimum the number of methods an object must implement. The API is designed to adhere to the following broad principles:

**Consistency:** All objects (basic or composite) share a consistent interface composed of a limited set of methods. This interface is documented in a consistent manner for all objects.

**Inspection:** Constructor parameters and parameter values determined by learning algorithms are stored and exposed as public attributes.

**Non-proliferation of classes:** Learning algorithms are the only objects to be represented using custom classes. Datasets are represented as NumPy arrays or SciPy sparse matrices. Hyper-parameter names and values are represented as standard Python strings or numbers whenever possible. This keeps scikit learn easy to use and easy to combine with other libraries.

**Composition:** Many machine learning tasks are expressible as sequences or combinations of transformations to data. Some learning algorithms are also naturally viewed as meta-algorithms parameterized on other algorithms. Whenever feasible, such algorithms are implemented and composed from existing building blocks. Sensible defaults. Whenever an operation requires a user-defined parameter, an appropriate default value is defined by the library. The default value should cause the operation to be performed in a sensible way (giving a baseline solution for the task at hand).

**Components of scikit-learn:**

Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

**Supervised learning algorithms:**Think of any supervised learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are **part of** scikit-learn toolbox. The spread of algorithms is one of the big reasons for high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.

**Cross-validation:**There are various methods to check the accuracy of supervised models on unseen data.

**Unsupervised learning algorithms:**Again there is a large spread of algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.

**Various toy datasets:**This came in handy while learning scikit-learn. I had learnt SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.

**Feature extraction:**Useful for extracting features from images and text (e.g. Bag of words).

**Estimators**

The estimator interface is at the core of the library. It defines instantiation mechanisms of objects and exposes a fit method for learning a model from training data. All supervised and unsupervised learning algorithms (e.g., for classification, regression or clustering) are offered as objects implementing this interface. Machine learning tasks like feature extraction, feature selection or dimensionality reduction are also provided as estimators.

**Predictor**

The predictor interface extends the notion of an estimator by adding a predict method that takes an array X test and produces predictions for X test, based on the learned parameters of the estimator (we call the input to predict “X test” in order to emphasize that predict generalizes to new data). In the case of supervised learning estimators, this method typically returns the predicted labels or values computed by the model.

**Transformers**

Since it is common to modify or filter data before feeding it to a learning algorithm, some estimators in the library implement a transformer interface which defines a transform method. It takes as input some new data X test and yields as output a **transformed version of X test.** Pre-processing, feature selection, feature extraction and dimensionality reduction algorithms are all provided as transformers within the library.

**Advanced API**

The advanced API mechanisms for building meta-estimators, composing complex estimators and selecting models

**Meta-estimators**

Some machine learning algorithms are expressed naturally as meta-algorithms parameterized on simpler algorithms. . Examples include ensemble methods which build and combine several simpler models (e.g., decision trees), or multiclass and multi label classification schemes which can be used to turn a binary classifier into a multiclass or multi label classifier.

**Pipelines and feature unions**

A distinguishing feature of the scikit-learn API is its ability to compose new estimators from several base estimators. Composition mechanisms can be used to combine typical machine learning workflows into a single object which is itself an estimator, and can be employed wherever usual estimators can be used. **Extending scikit-learn** To ease code reuse, simplify implementation and skip the introduction of superfluous classes, the Python principle of duck typing is exploited throughout the code base.

**CHAPTER-4**

**SYSTEM ARCHITECTURE AND DESIGN**

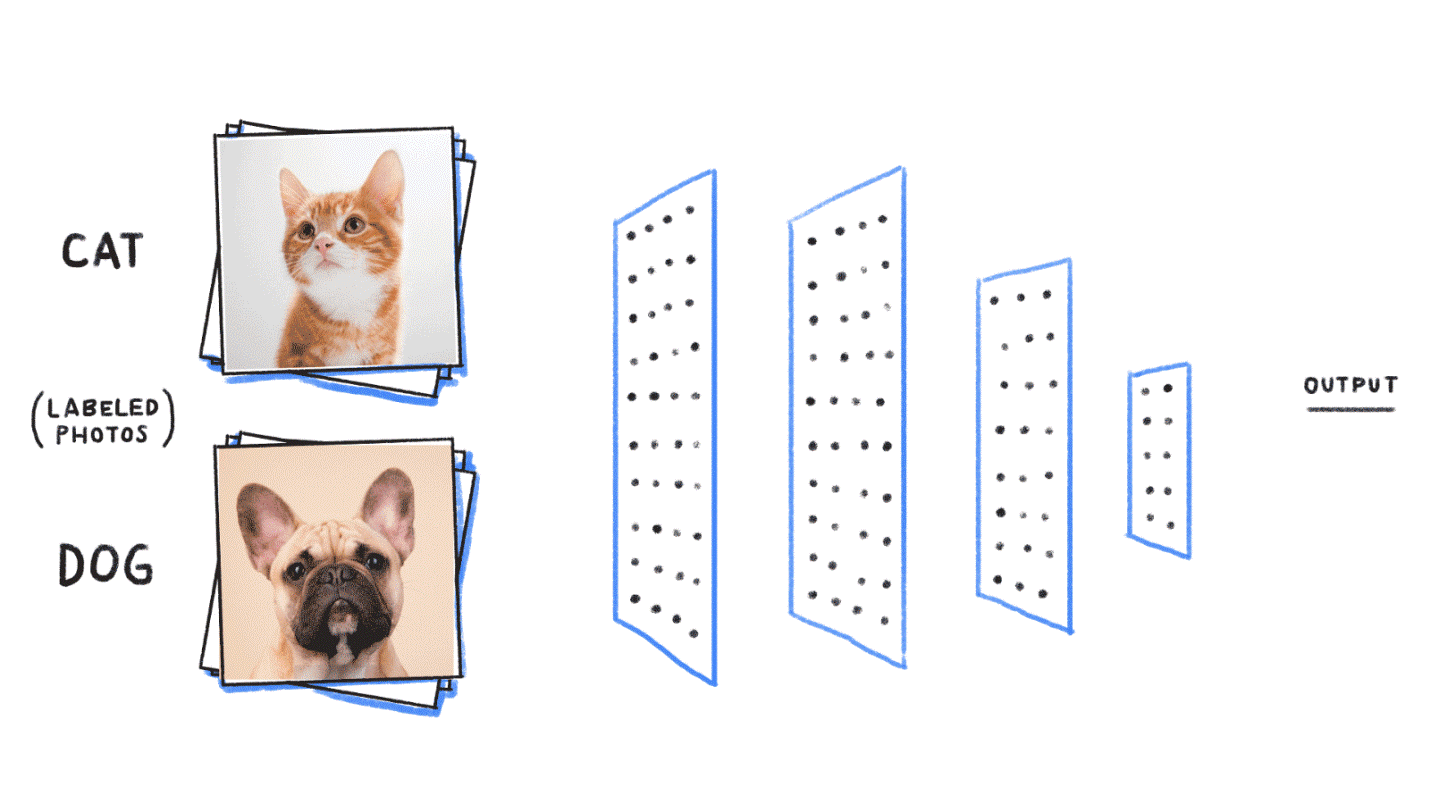
**Modules:**

**Image Acquisition and Pre-processing**

**Data Preparation and Model construction**

**Model training**

**Model testing and evaluation**

****

**Image Acquisition and Pre-processing**

In this module we will get the data from the online source. Further we will resize the image for future use. Image resizing, or image scaling, is a geometric image transformation which modifies the image size based on an image interpolation algorithm. This image scaling process can increase or decrease the resolution of a target image so that the absolute size of image data is adjusted.

Computers are able to perform computations on numbers and are unable to interpret images in the way that we do. We have to somehow convert the images to numbers for the computer to understand. The image will be converted to grayscale (range of gray shades from white to black) the computer will assign each pixel a value based on how dark it is. All the numbers are put into an array and the computer does computations on that array. We then feed the resulting array for next step.

**Data Preparation and Model construction**

Many a times, people first split their dataset into 2 — Train and Test. After this, they keep aside the Test set, and randomly choose X% of their Train dataset to be the actual Train set and the remaining (100-X)% to be the Validation set, where X is a fixed number(say 80%), the model is then iteratively trained and validated on these different sets. So we will follow the same method to prepare data for training and testing phase.

We are building our model by using Convolutional neural network. Convolutional neural networks (CNN) are a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. Now that we’re done pre-processing, we can start implementing our neural network. We’re going to have 3 convolution layers with 2 x 2 max-pooling.

Max-pooling: A technique used to reduce the dimensions of an image by taking the maximum pixel value of a grid. This also helps reduce over fitting and makes the model more generic. After that, we add 2 fully connected layers. Since the input of fully connected layers should be two dimensional, and the output of convolution layer is four dimensional, we need a flattening layer between them. At the very end of the fully connected layers is a SoftMax layer.

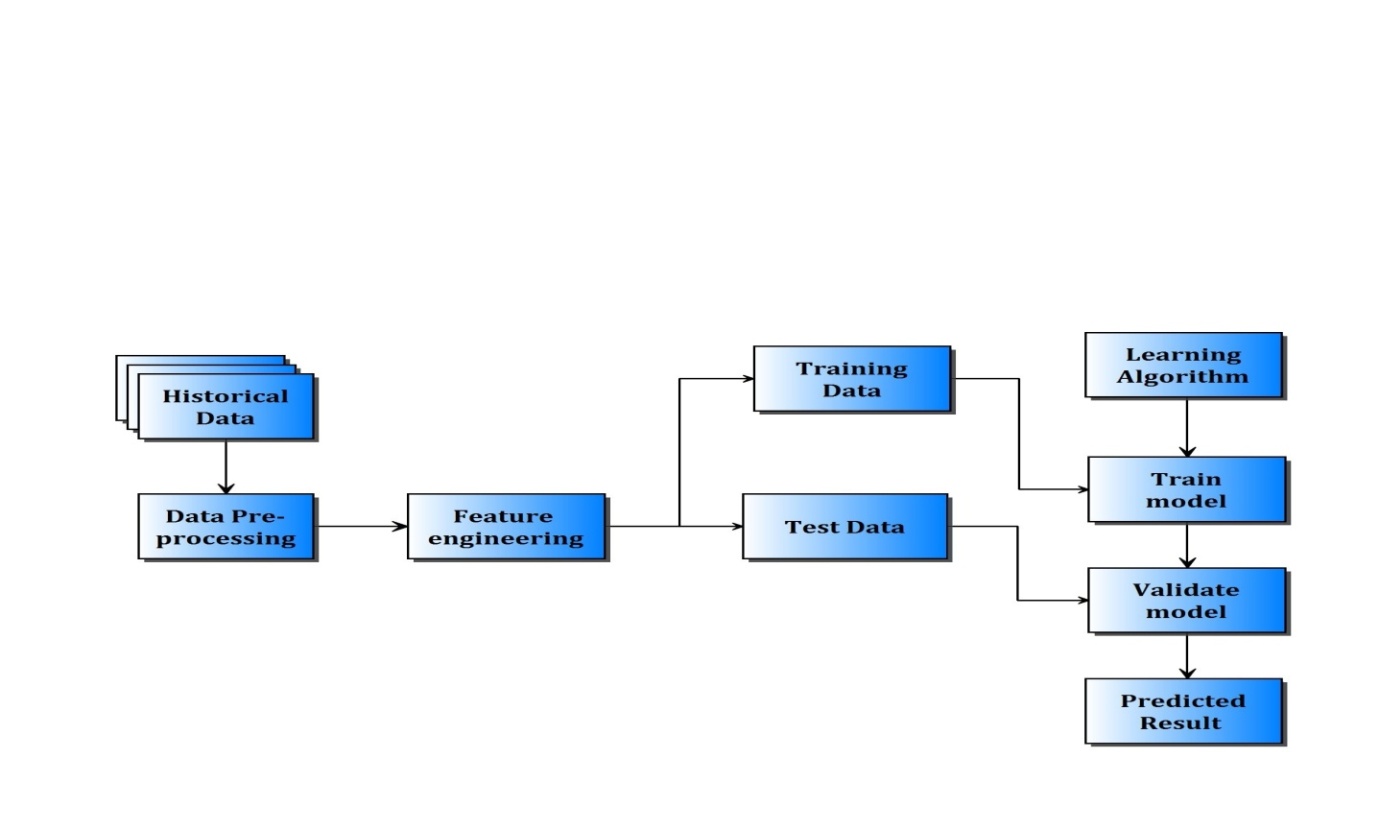
**Model training**

After model construction it is time for **model training.** We were able to build an artificial convolutional neural network that can recognize images. Split the dataset into train and test dataset. Finally we will build and train the model using training dataset.

**Model testing and evaluation**

Once the model has been trained it is possible to carry out **model testing.** During this phase a test set of data is loaded. This data set has never been seen by the model and therefore its true accuracy will be verified. Finally, the saved model can be used in the real world. The name of this phase is **model evaluation**. This means that the model can be used to evaluate new data.

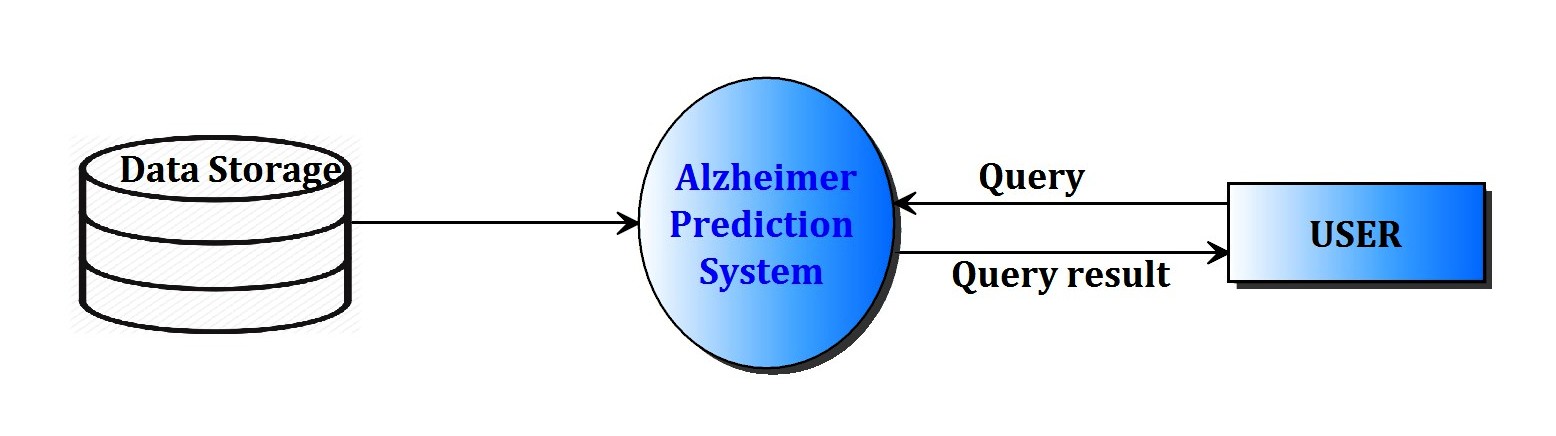
**SYSTEM ARCHITECTURE:** A system architecture diagram would be used to show the relationship between different components. Usually they are created for systems which include hardware and software and these are represented in the diagram to show the interaction between them.

****

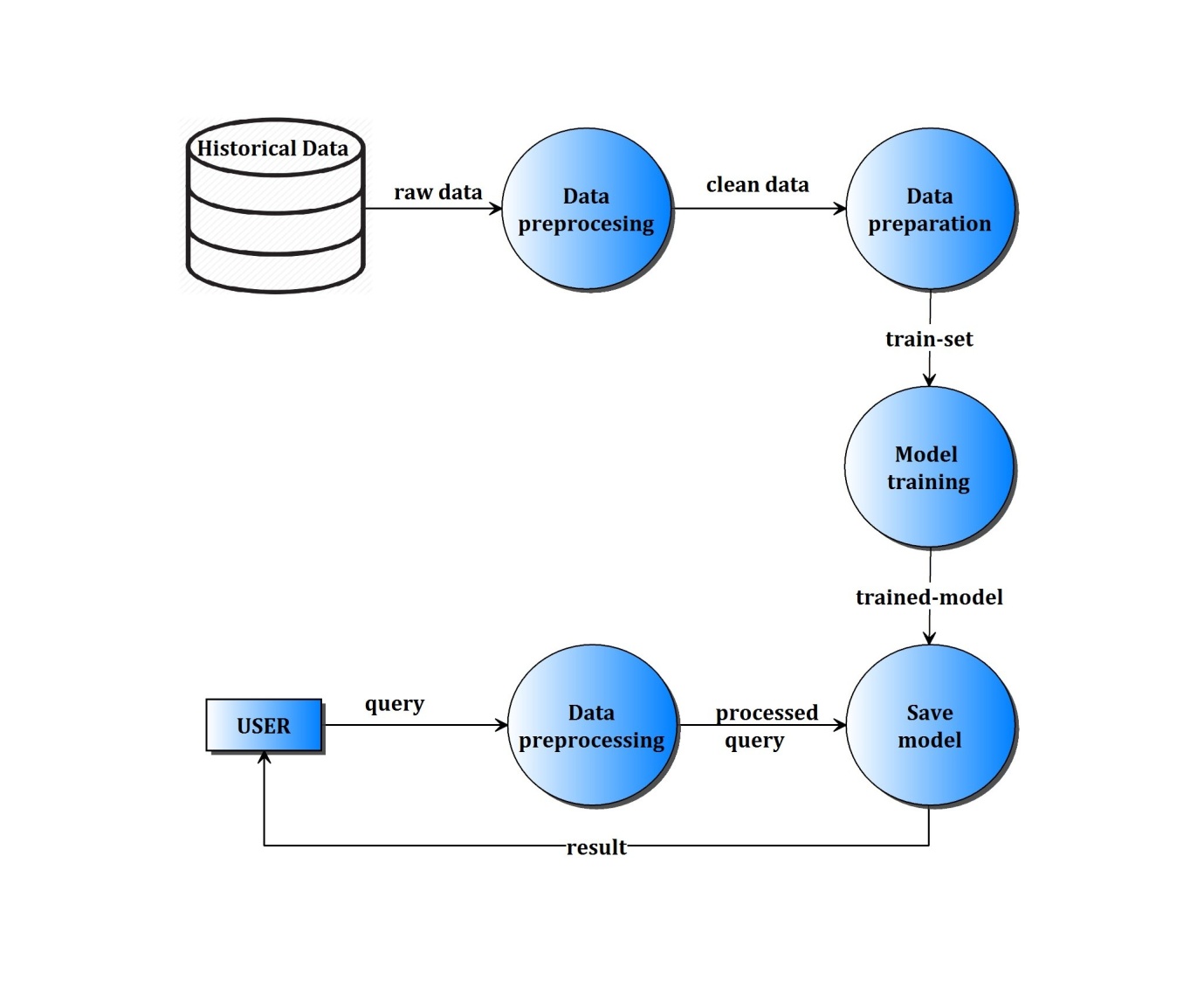
**Figure 4.1 system architecture**

**DATA FLOW DIAGRAMS:** A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling 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.

**DAFTAFLOW DIAGRAM LEVEL O**

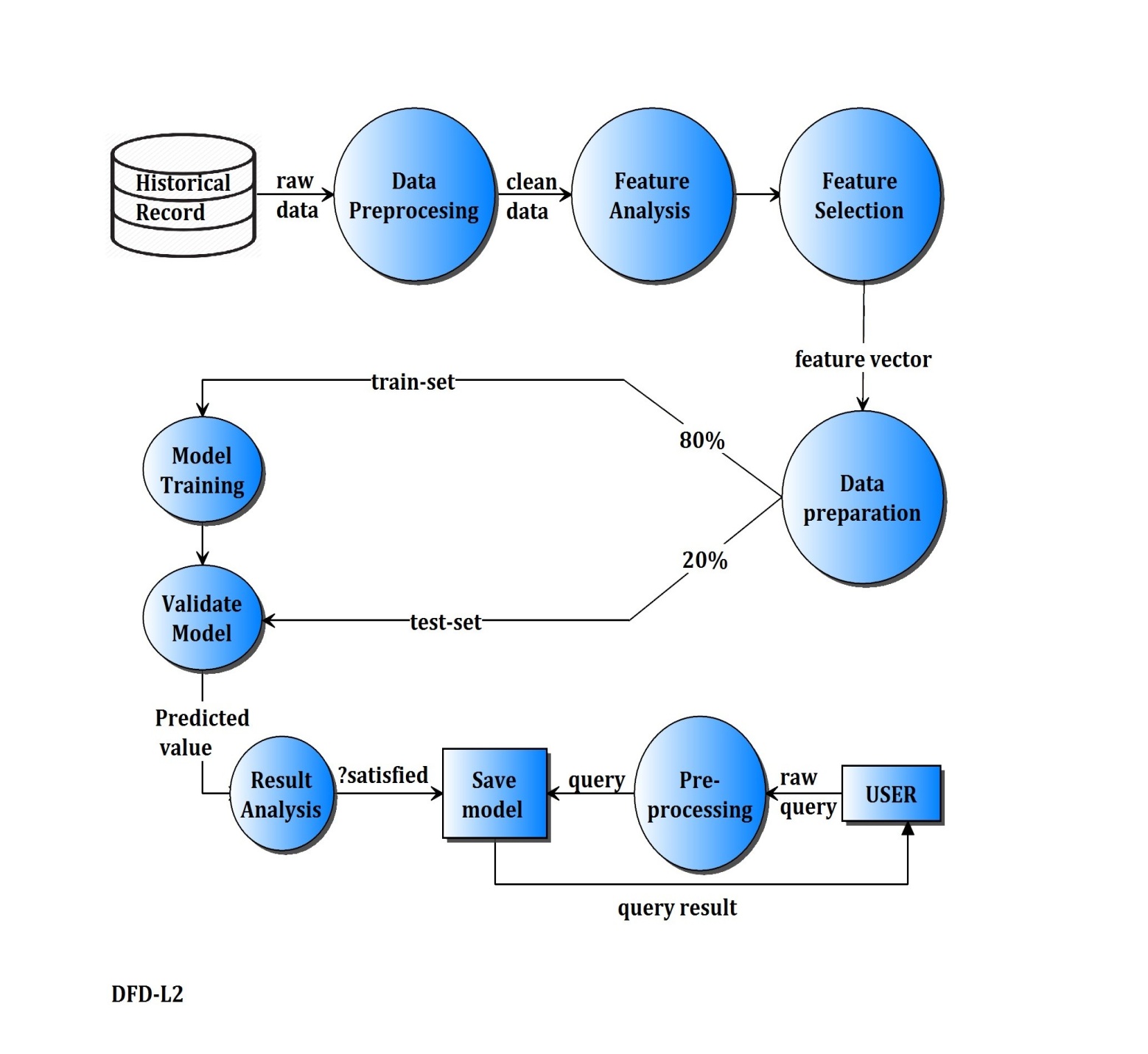
**Figure 4.2 Dataflow diagram level 0**

**DAFTAFLOW DIAGRAM LEVEL 1**

****

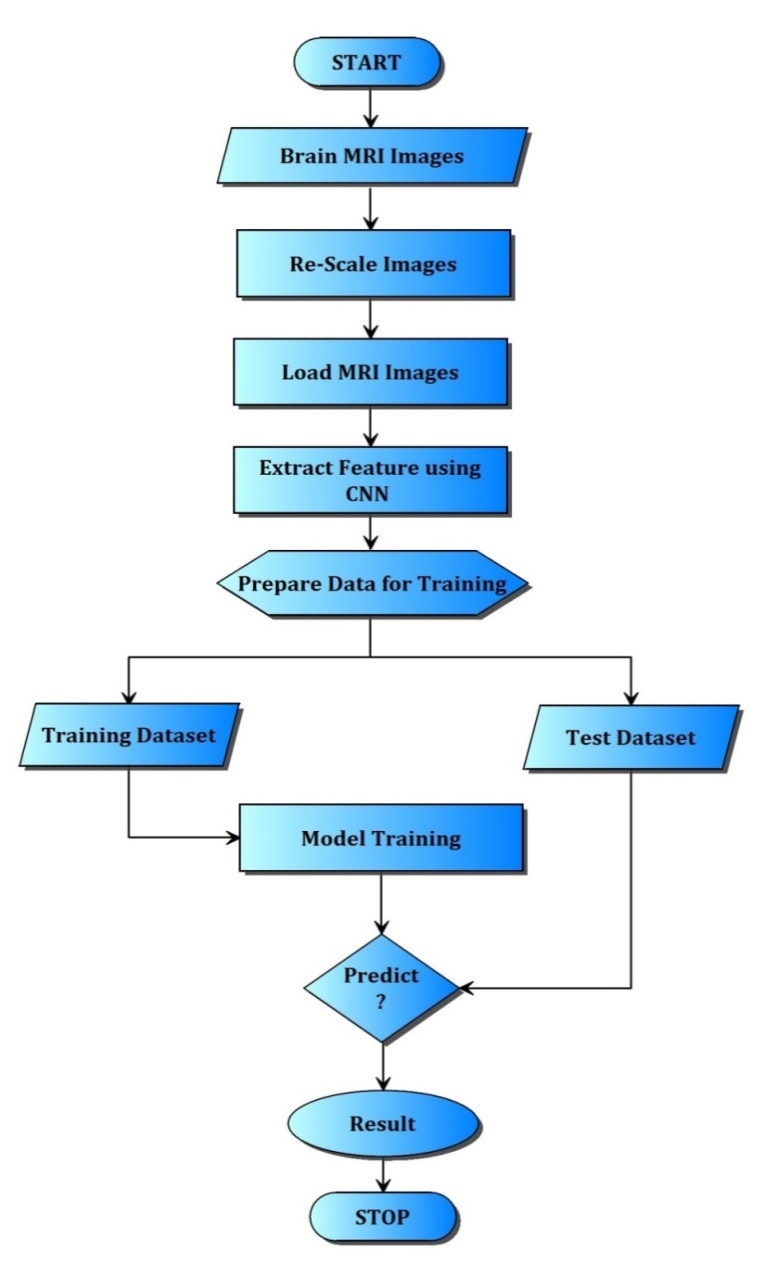
**Figure 4.3 Dataflow diagram level 1**

**DAFTAFLOW DIAGRAM LEVEL 2**

****

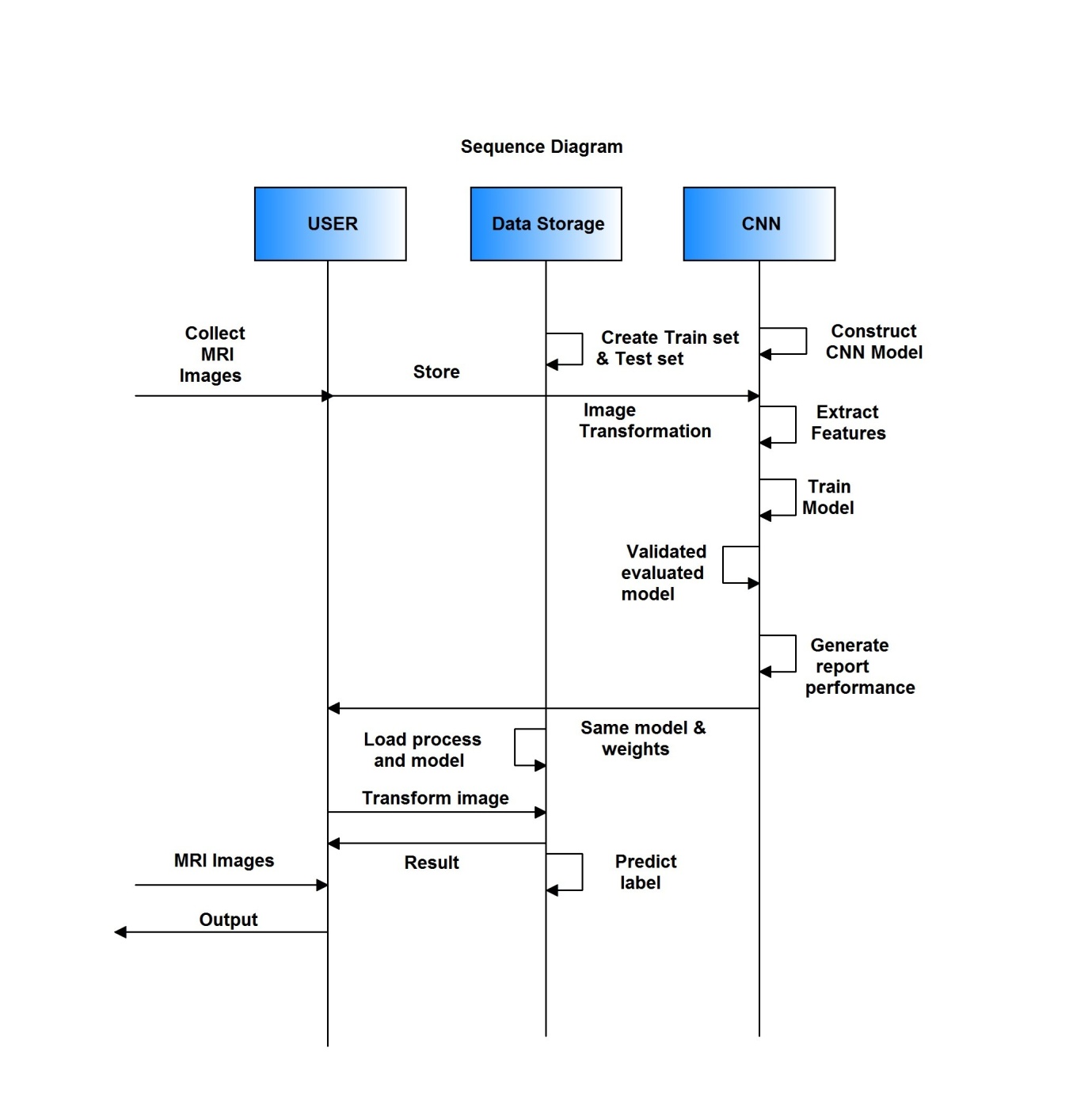
**Figure 4.4 Dataflow diagram level 2**

**FLOWCHART:** A [flowchart](https://project-management-knowledge.com/definitions/f/flowchart/) is one of the seven basic quality tools used in project management and it displays the actions that are necessary to meet the goals of a particular task in the most practical sequence. Also called as process maps, this type of tool displays a series of steps with branching possibilities that depict one or more inputs and transforms them to outputs.

****

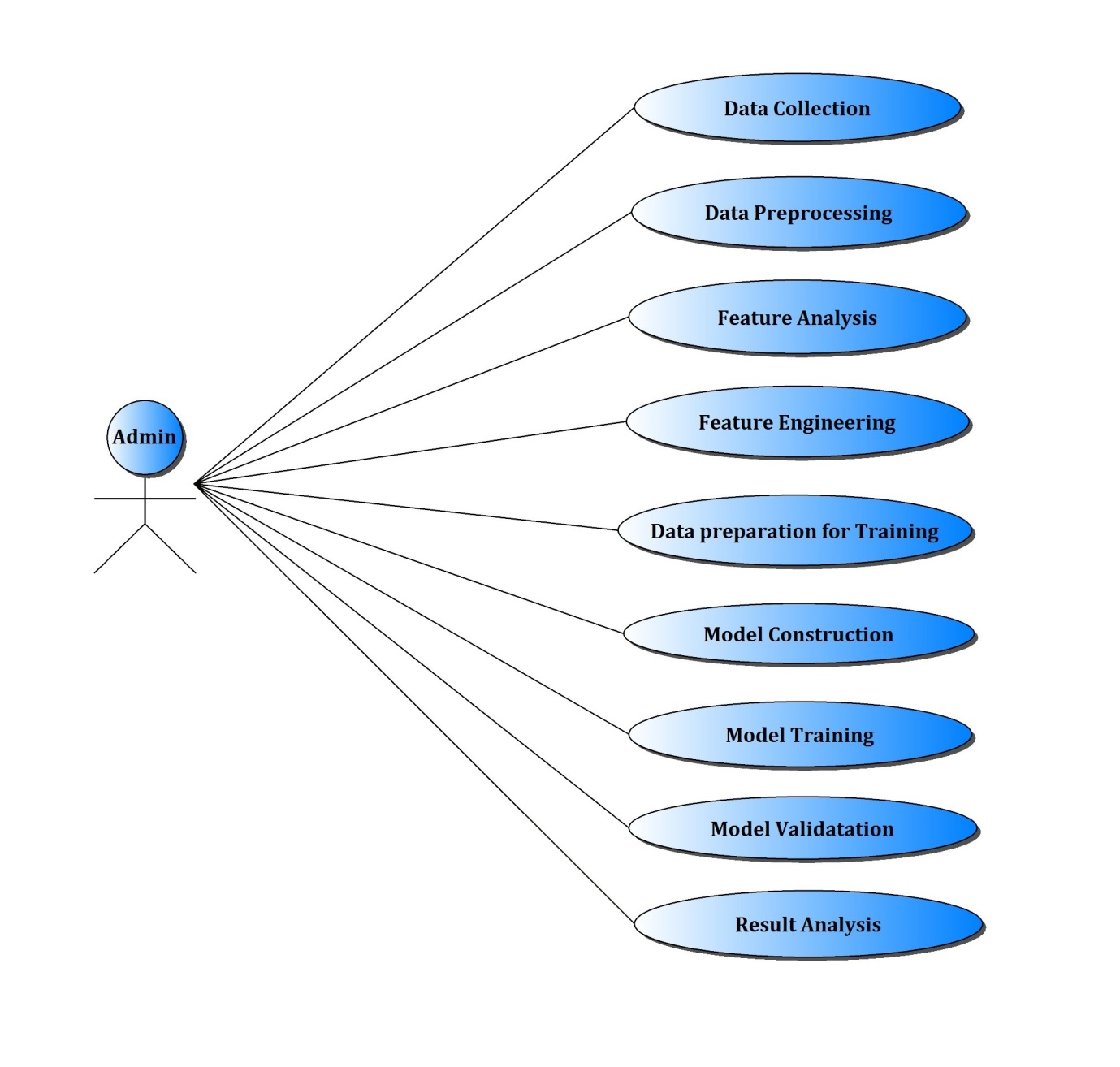
**Figure 4.5 Flowchart**

**SEQUENCE DIAGRAM:** A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

**Figure 4.6 Sequence diagram**

**USECASE DIAGRAM**: A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

**USECASE DIAGRAM-USER**

****

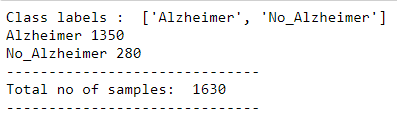
**Figure 4.7 Use case diagram**

**CHAPTER-5**

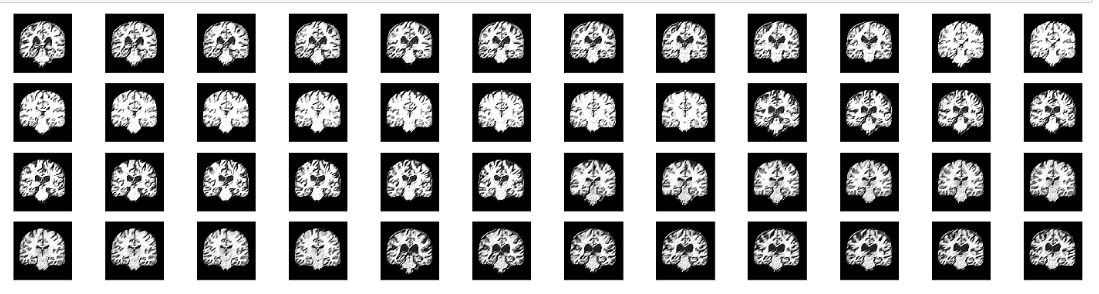
**RESULT AND CONCLUSION**

**Result:**

**Fig 5.1- Input**

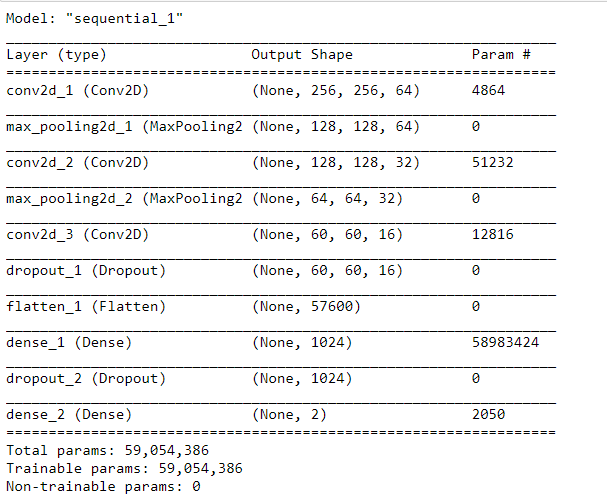
****

**Fig 5.2 – Images after applying Gabor filter ( Pre-processing stage)**



The above figure shows the images after applying the Gabor filter. The Gabor filter has been applied twice for better feature extraction which helps in Data modelling. Applying the Gabor filter twice has increased the accuracy.

**Fig 5.3 – Sequential steps for Data Modelling**

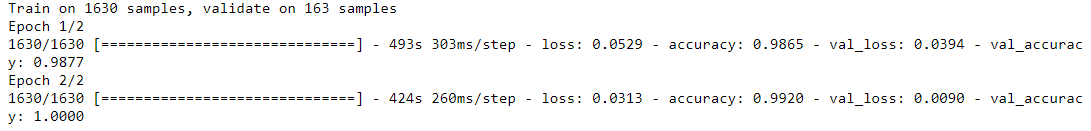


The above figure shows the steps of data modelling in sequential order. Multiple layers are used like convolution-2D , Max Pooling, Dense, Dropout and Flatten.

Param # denotes the number of parameters.

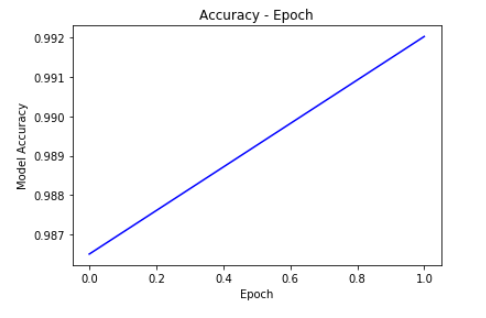
Total parameters are 59,054,386 in which all the parameters are used for training.

**Fig 5.4 – Training Phase and it’s accuracy**



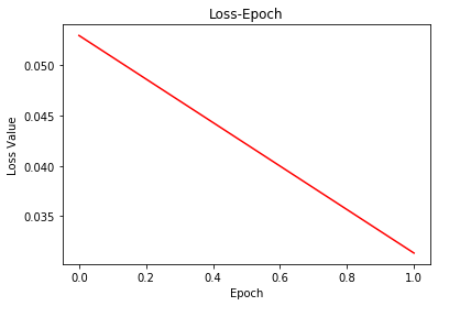
The above figure shows the Training phase accuracy which is found out to be 99.20%.

**Fig 5.5 – Accuracy epoch**



The above graph shows the accuracy which is found out to be 99.20%.

**Fig 5.6 – Loss epoch**



The above graph shows the loss curve which is found out to be 0.8%.

**Fig 5.7 – Confusion matrix**

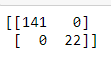
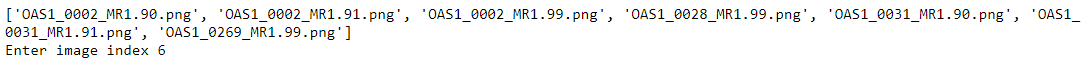


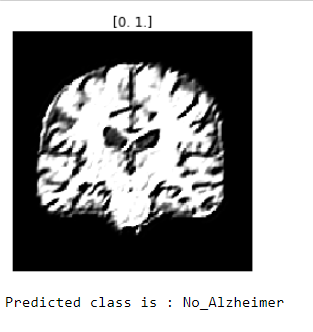
Fig 5.7 represents the confusion matrix from which Accuracy, recall, precision etc. can be found.

**Fig 5.8 – Testing phase**



The above figure denotes the input that we give to the testing phase. Here, a sample of 7 images is taken into consideration. These 7 images are given indices from 0 to 6. We can enter the index of the image for which we wish to predict the output i.e., if that person is affected with Alzheimer’s or not. Here, we have selected the index to be 6. The result is displayed in Fig-5.9 as “Alzheimer” or “No\_Alzheimer” accordingly.

**Fig 5.9- The Final Result**



The above figure shows the final output. It shows if the selected input image has Alzheimer’s or not. Here, in this scenario, the image that we had selected is not affected with Alzheimer’s disease. Hence, it is displayed as “No\_Alzheimer”.

**Conclusion:**

This system comes under deep learning which is advanced technique at present. CNN is more suitable for image processing especially in image classification. The Gabor filter has been used twice from which we were able to extract multiple features of the image. The confusion matrix has also been generated from which we can find various parameters like Accuracy, Precision, Recall, f1 etc. The Accuracy Epoch and Loss Epoch have been plotted. We conclude the experimental result what we are getting from developed system is more than 95% Accurate.

**References:**

1. R. Ju, C. Hu, p. zhou and Q. Li, "Early Diagnosis of Alzheimer's Disease Based on Resting-State Brain Networks and Deep Learning," in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 1, pp. 244-257, 1 Jan.-Feb. 2019.
2. R. C. Petersen, et al., “Current concepts in mild cognitive impairment,” Archives Neurol., vol. 58, no. 12, pp. 1985–1992, 2001.
3. T. Kam, H. Zhang, Z. Jiao and D. Shen, "Deep Learning of Static and Dynamic Brain Functional Networks for Early MCI Detection," in *IEEE Transactions on Medical Imaging*, vol. 39, no. 2, pp. 478-487, Feb. 2020.
4. M. Wang, C. Lian, D. Yao, D. Zhang, M. Liu and D. Shen, "Spatial-Temporal Dependency Modeling and Network Hub Detection for Functional MRI Analysis via Convolutional-Recurrent Network," in IEEE Transactions on Biomedical Engineering.
5. H. Fuse, K. Oishi, N. Maikusa, T. Fukami and J. A. D. N. Initiative, "Detection of Alzheimer's Disease with Shape Analysis of MRI Images," *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, Toyama, Japan, 2018
6. S. Matoug, A. Abdel-Dayem, K. Passi, W. Gross and M. Alqarni, "Predicting Alzheimer's disease by classifying 30Brain MRI images using SVM and other well-defined classifiers ",IOPjournal publication,2011
7. B. AI-Naami , N. Gharaibeh, and A. AIRazzaq Kheshman, "Automated Detection of Alzheimer Disease Using Region Growing technique and Artificial Neural Network" , ELSEVIER journal publication,2013.
8. K. R. Graya, P. Aljabara, R. A. Heckemannc, A. Hammersc, and D. Rueckerta, “Random forest-based similarity measures for multi-modal classiﬁcation of Alzheimer’s disease,” Neuroimage, vol. 65, pp. 167–175, 2013.