**Diagnosis of Alzheimer’s Disease Using Convolutional Neural Network with Select Slices by Landmark on Hippocampus in MRI Images**

**INTRODUCTION**

Alzheimer’s disease (AD) is a major public health priority [1]. Globally, around 44 million people have been diagnosed The associate editor coordinating the review of this manuscript and approving it for publication was Jerry Chun-Wei Lin . with AD globally, which may reach 131.5 million people by 2050 [2]. AD is a progressive, neurodegenerative disease that affects elderly people over 65 years old and impacts memory and cognitive function [3]. Although there is no known cure for AD, some medications and treatments can temporarily relieve symptoms or slow down AD progression [4]. Research 61688 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/ VOLUME 11, 2023 Y. Pusparani et al.: Diagnosis of AD Using CNN With Select Slices by Landmark on Hippocampus studies have demonstrated that early AD diagnosis can make a living with the disease easier [3]. In an early AD diagnosis, observing and exploring the deterioration process in the brain regions is important before the progression of the disease. Hippocampus is one of the most affected areas of the brain and is easily accessible as a biomarker of AD [5], [6]. For example, the degeneration of cholinergic circuits in the hippocampus and reduced volume changes in the hippocampus are related to memory loss [5]. In addition, a severe volume reduction of the hippocampus can be easily detected using Magnetic Resonance Imaging (MRI) images and is widely used for diagnosing AD [7]. MRI has become an excellent and valuable tool with a highly effective imagery technique for diagnosing and analyzing structural changes in the brain [8]. Moreover, structural MRI image biomarkers are used in three categories: AD, Mild Cognitive Impairment (MCI), and Normal Control (NC) [9] Three categories of AD are used to understand subtle changes in disease progression on an MRI image in the early stages of AD [10]. Recently, machine learning models have been developed to diagnose AD based on MRI images [11]. The advances in machine learning have the potential to classify complex patterns from MRI images, and the diagnosis can be finalized in a brief time [12]. For example, a study by Kazemi and Houghten used machine learning models to classify different categories of AD [13]. Other studies showed that cancer detection accuracy is comparable to manual detection. Therefore, machine learning can reduce the time and is expected to perform consistently in large amounts of data at any time. In contrast, manual diagnosis results may be affected the time to read the MRI images in diagnosing AD. Furthermore, with the advantages, machine learning has become the preferred method for medical image classification [14]. In terms of classification tasks in different AD stages, first, use binary classification between two categories to classify AD, such as AD vs. NC, MCI vs. AD, and NC vs. MCI [15]. In comparison, some studies used the multiclass classification to classify AD in three categories [16]. In addition, a study by Kazemi and Houghten used machine learning to classify AD in three categories (i.e., AD, MCI, and NC) [13]. The use of multiclass classification can be beneficial in distinguishing the results of three categories of AD because the binary classification consists of only two categories, whereas the multiclass classification consists of more than two categories. Thus, the multiclass classification can help improve the clinical decisions on whether someone will develop the disease through the results of each category to diagnose AD. In order to improve performance for AD classification, several studies have proposed different methods to improve the accuracy performance in machine learning using magnetic resonance imaging (MRI) images. For instance, the improvement of the MRI image quality to reduce the noises [17], the use of segmentation in a specific brain region [18], classification techniques with AdaBoost [19], and slice-based method [20]. In general, the entire slices of MRI images are composed of 100 to 250 slices [21], [22]. However, Lopez et al. found that the results in classification improve the accuracy with several slices than using the entire slices [23]. In spite of accuracy improvement for AD classification, there is no detailed information on how they select several slices in MRI images within three views and three categories. For example, in a study by Kang et al., they selected 11 slices based on the highest classification accuracy in coronal view between 1 to 145 slices [24]. Thus, there is no information about selecting several slices regarding the early diagnosis of AD in the hippocampus using MRI images. Furthermore, getting the information on the biomarker for early AD diagnosis (i.e., hippocampus) in MRI images to select several slices requires medical experts’ knowledge and experiences [25], [26]. The medical experts’ information can be used as the ground truth in three views and three categories for AD classification. Still, the three views of MRI images are commonly known as axial, coronal, and sagittal [27]. Using three views of MRI images may offer complementary features useful for AD classification. Even though MRI images can provide the landmark of the hippocampus region in three views, medical experts usually require one view, which will be beneficial for diagnosing AD. Thus, the selecting slices method in MRI images can be used for AD diagnosis and provide more computational simplicity than using entire slices in three views and three categories (i.e., AD, MCI, and NC). Finally, the selecting slices method focuses on the landmarks in the hippocampus region in MRI images and is used for AD classification. In addition, working on the hippocampus region on MRI images can involve more advantages to improve the performances of AD classification. Therefore, we hypothesized that the selecting slices method of MRI images using landmarks in the hippocampus region might improve performance in classifying AD. In order to validate our proposed method, we compare the classification result with the use of entire slices in MRI images. With the proposed method, this study aims to see which views (i.e., axial, coronal, and sagittal) of MRI images are higher accuracy for AD classification in machine learning. Then, we used multiclass classification on MRI images to get the result in three categories (i.e., AD, MCI, and NC) for AD classification.

* 1. **Objective of the project:**

Alzheimer’s disease (AD) is a major public health priority. Hippocampus is one of the most affected areas of the brain and is easily accessible as a biomarker using MRI images in machine learning for diagnosing AD. In machine learning, using entire MRI image slices showed lower accuracy for AD classification. We present the select slices method by landmarks on the hippocampus region in MRI images. This study aims to see which views of MRI images have higher accuracy for AD classification. Then, to get the value of three views and categories, we used multiclass classification with the publicly available Alzheimer’s Disease Neuroimaging Initiative (ADNI) dataset using Resnet50 and LeNet. The models were used in a total dataset of 4,500 MRI slices in three views and categories. Our study demonstrated that the selecting slices performed better than using entire slices in MRI images for AD classification. Our method improves the accuracy of machine learning, and the coronal view showed higher accuracy. This method played a significant role in improving the accuracy of machine learning performance. The results for the coronal view were similar to the medical experts usually used to diagnose AD. We also found that LeNet models became the potential model for AD classification.

**2. LITERATURE SURVEY:**

**The clinical use of structural MRI in Alzheimer disease**

Structural imaging based on magnetic resonance is an integral part of the clinical assessment of patients with suspected Alzheimer dementia. Prospective data on the natural history of change in structural markers from preclinical to overt stages of Alzheimer disease are radically changing how the disease is conceptualized, and will influence its future diagnosis and treatment. Atrophy of medial temporal structures is now considered to be a valid diagnostic marker at the mild cognitive impairment stage. Structural imaging is also included in diagnostic criteria for the most prevalent non-Alzheimer dementias, reflecting its value in differential diagnosis. In addition, rates of whole-brain and hippocampal atrophy are sensitive markers of neurodegeneration, and are increasingly used as outcome measures in trials of potentially disease-modifying therapies. Large multicenter studies are currently investigating the value of other imaging and nonimaging markers as adjuncts to clinical assessment in diagnosis and monitoring of progression. The utility of structural imaging and other markers will be increased by standardization of acquisition and analysis methods, and by development of robust algorithms for automated assessment.

**Impact of new drugs for therapeutic intervention in Alzheimer’s disease**

The increases in population ageing and growth are leading to a boosting in the number of people living with dementia, Alzheimer's disease (AD) being the most common cause. In spite of decades of intensive research, no cure for AD has been found yet. However, some treatments that may change disease progression and help control symptoms have been proposed. Beyond the classical hypotheses of AD etiopathogenesis, i.e., amyloid beta peptide (Aβ) accumulation and tau hyperphosphorylation, a trend in attributing a key role to other molecular mechanisms is prompting the study of different therapeutic targets. Hence, drugs designed to modulate inflammation, insulin resistance, synapses, neurogenesis, cardiovascular factors and dysbiosis are shaping a new horizon in AD treatment. Within this frame, an increase in the number of candidate drugs for disease modification treatments is expected, as well as a focus on potential combinatory multidrug strategies.The present review summarizes the latest advances in drugs targeting Aβ and tau as major contributors to AD pathophysiology. In addition, it introduces the most important drugs in clinical studies targeting alternative mechanisms thought to be involved in AD's neurodegenerative process.

**Hippocampus and its involvement in Alzheimer’s disease: A review**

Hippocampus is the significant component of the limbic lobe, which is further subdivided into the dentate gyrus and parts of Cornu Ammonis. It is the crucial region for learning and memory; its sub-regions aid in the generation of episodic memory. However, the hippocampus is one of the brain areas affected by Alzheimer’s (AD). In the early stages of AD, the hippocampus shows rapid loss of its tissue, which is associated with the functional disconnection with other parts of the brain. In the progression of AD, atrophy of medial temporal and hippocampal regions are the structural markers in magnetic resonance imaging (MRI). Lack of sirtuin (SIRT) expression in the hippocampal neurons will impair cognitive function, including recent memory and spatial learning. Proliferation, differentiation, and migrations are the steps involved in adult neurogenesis. The microglia in the hippocampal region are more immunologically active than the other regions of the brain. Intrinsic factors like hormones, glia, and vascular nourishment are instrumental in the neural stem cell (NSC) functions by maintaining the brain’s microenvironment. Along with the intrinsic factors, many extrinsic factors like dietary intake and physical activity may also influence the NSCs. Hence, pro-neurogenic lifestyle could delay neurodegeneration.

**MRI of hippocampus in incipient Alzheimer’s disease**

Alzheimer's disease (AD) is the most common cause of dementia, yet impossible to diagnose precisely without invasive techniques, particularly at the onset of the disease. Therefore, a reliable diagnostic method is needed. The hippocampus is a part of the mesial temporal lobe memory system, and known to be affected early in the course of AD. Recent development of imaging techniques, particularly magnetic resonance imaging (MRI), has made the evaluation of diminutive brain structures, such as the hippocampus, conceivable. The purpose of this study was to focus on the sensitivity and specificity of different approaches of hippocampal imaging by MRI, and their applications for the diagnosis of incipient AD. Hippocampal pathology was evaluated by means of linear (interuncal distance, IUD), planimetric (hippocampal area) and volumetric measurements, complete with T2 relaxometry using an 1.5 T imager. The accuracy of hippocampal measurements was compared to that of the amygdala and the frontal lobes. Various procedures for normalization of the data to the head and brain size were compared.

**Prediction of Alzheimer’s disease dementia with MRI beyond the shortterm: Implications for the design of predictive models**

Magnetic resonance imaging (MRI) volumetric measures have become a standard tool for the detection of incipient Alzheimer's Disease (AD) dementia in mild cognitive impairment (MCI). Focused on providing an earlier and more accurate diagnosis, sophisticated MRI machine learning algorithms have been developed over the recent years, most of them learning their non-disease patterns from MCI that remained stable over 2–3 years. In this work, we analyzed whether these stable MCI over short-term periods are actually appropriate training examples of non-disease patterns. To this aim, we compared the diagnosis of MCI patients at 2 and 5 years of follow-up and investigated its impact on the predictive performance of baseline volumetric MRI measures primarily involved in AD, i.e., hippocampal and entorhinal cortex volumes. Predictive power was evaluated in terms of the area under the ROC curve (AUC), sensitivity, and specificity in a trial sample of 248 MCI patients followed-up over 5 years.

**Automated detection, selection and classification of hippocampal landmark points for the diagnosis of Alzheimer’s disease**

Alzheimer’s disease (AD) is a neurodegenerative, progressive, and irreversible disease that accounts for up to 80% of all dementia cases. AD predominantly affects older adults, and its clinical diagnosis is a challenging evaluation process, with imprecision rates between 12 and 23%. Structural magnetic resonance (MR) imaging has been widely used in studies related to AD because this technique provides images with excellent anatomical details and information about structural changes induced by the disease in the brain. Current studies are focused on detecting AD in its initial stage, i.e., [mild cognitive impairment](https://www.sciencedirect.com/topics/medicine-and-dentistry/mild-cognitive-impairment) (MCI), since [treatments](https://www.sciencedirect.com/topics/medicine-and-dentistry/therapeutic-procedure) for preventing or delaying the onset of symptoms is more effective when administered at the early stages of the disease. This study proposes a new technique to perform MR [image classification](https://www.sciencedirect.com/topics/computer-science/image-classification) in AD diagnosis using discriminative hippocampal point landmarks among the cognitively normal (CN), MCI, and AD populations.

**Classification of structural MRI for detecting Alzheimer’s disease**

Alzheimer’s Disease (AD) is a pathological form of dementia that degenerates brain structures. AD affects millions of elderly people over the world and the number of people with AD doubles every year. Detecting AD years before the effects of disease using structural magnetic resonance imaging (MRI) of the brain is possible. Neuroimaging features that are extracted from the structural brain MRI can be used to predict AD by revealing disease related patterns. Machine learning techniques can detect AD and predict conversions from mild cognitive impairment (MCI) to AD automatically and successfully by using these neuroimaging features. In this study common structural brain measures such as volumes and thickness of anatomical structures that are obtained from The Open Access Series of Imaging Studies (OASIS) and made publicly available by https://www.nmr.mgh.harvard.edu/lab/mripredict are analysed. State-of-the-art machine learning techniques, namely support vector machines (SVM), k-nearest neighbour (kNN) algorithm and backpropagation neural network (BP-NN) are employed to discriminate AD and mild AD from healthy controls. Training hyperparameters of the classifiers are tuned using classification accuracy which is obtained with 5-fold cross validation. Prediction performance of the techniques are compared using accuracy, sensitivity and specificity. Results of the system revealed that AD can be distinguished from the healthy controls successfully using multivariate morphological features and machine learning tools. According to the performed experiments SVM is the most successful classifier for detecting AD with classification accuracies up to 82%.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

In Existing System, using machine learning algorithms such as Random Forest Algorithm, Naïve Bayes Algorithm. By using this machine learning algorithms accuracy is less.

**DisAvantages of Existing System:**

1. Less Accuracy

**3.2 PROPOSED SYSTEM**

In proposed system, we are using deep learning algorithms such as RCNN, FRCNN, Yolov5, Yolov6 Algorithms. By using this algorithms, we can get high accuracy. In machine learning, using entire MRI image slices showed lower accuracy for AD classification. We present the select slices method by landmarks on the hippocampus region in MRI images. This study aims to see which views of MRI images have higher accuracy for AD classification.

**Advantages of Proposed system:**

1. High Accuracy

**Modules Information:**

To implement this project we have used same dataset given in your requirement file and to implement this project we have designed following modules

**FUNCTIONAL REQUIREMENTS:**

**SOFTWARE REQIREMENTS:**

**System Atributes:**

1. Filename
2. dataset
3. X, Y, mse, X\_train, X\_test, y\_train, y\_test

**Prototype:**

python 3.7.0 or 3.7.4

opencv-python==4.5.1.48

keras==2.3.1

tensorflow==1.14.0

protobuf==3.16.0

h5py==2.10.0

sklearn-extensions==0.0.2

scikit-learn==0.22.2.post1

Numpy

Pandas

**NON-FUNCTIONAL REQUIREMENT:**

**Usability:**  Usability is a quality attribute that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process.(how it was handle entire project easy)

**Security:** the quality or state of being secure: such as. a : freedom from danger : safety. b : freedom from fear or anxiety. c : freedom from the prospect of being laid off job security.

**Readability:** Readability is the ease with which a reader can understand a written text.

**Performance**: the execution of an action. : something accomplished : deed, feat. : the fulfillment of a claim, promise, or request : implementation. 3. : the action of representing a character in a play.

**Availability**: the quality or state of being available trying to improve the availability of affordable housing. 2 : an available person or thing.

**Scalability**: Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500 GB

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python (3.7.0)

**4. SYSTEM DESIGN**

**CLASS DIAGRAM:**

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake



**USECASE DIAGRAM:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as we



**SEQUENCE DIAGRAM**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.



**COLLABORATION DIAGRAM:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**COMPONENT DIAGRAM:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.



**DEPLOYMENT DIAGRAM:**

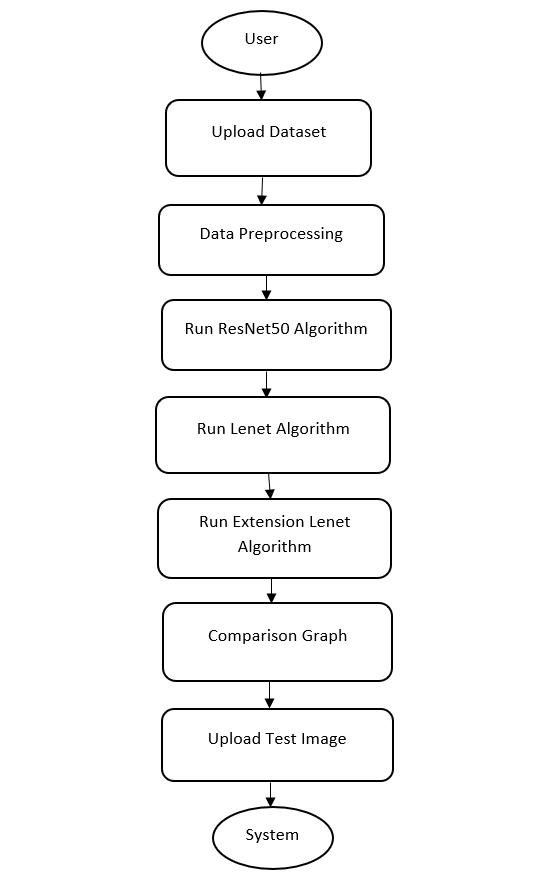
A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

****

**ACTIVITY DIAGRAM:**

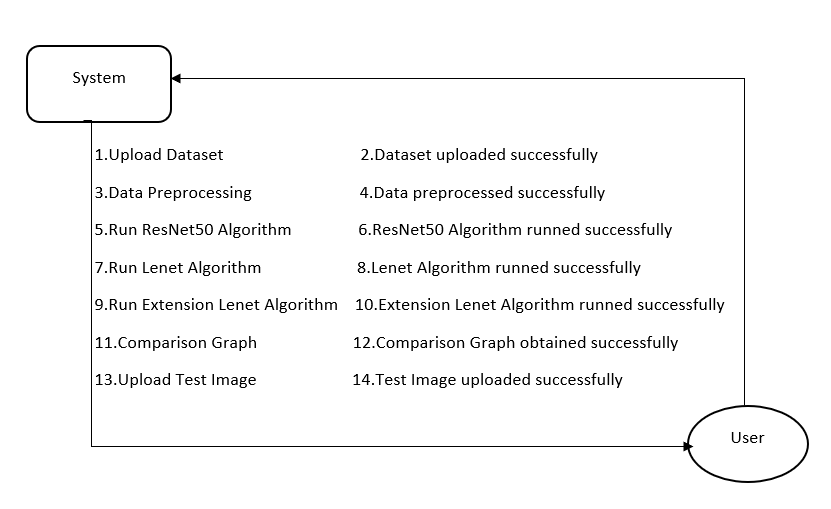
Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent



**Data flow:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.



**5. IMPLEMETATION**

**5.1 Python**

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

**History of Python:**

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

**Why Python was created?**

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

**Why the name Python?**

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

**Features of Python:**

**A simple language which is easier to learn**

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

If you are a newbie, it's a great choice to start your journey with Python.

**Free and open-source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code.

Python has a large community constantly improving it in each iteration.

**Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

**Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

**A high-level, interpreted language**

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

**Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQL dB library using import MySQL db.

Standard libraries in Python are well tested and used by hundreds of people. So, you can be sure that it won't break your application.

**Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

**Applications of Python:**

**1. Simple Elegant Syntax**

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

a = 2

b = 3

sum = a + b

print(sum)

**2. Not overly strict**

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

**3. Expressiveness of the language**

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

**4. Great Community and Support**

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

**5.2 Sample Code:**

"#import required libraries files\n",

"import cv2\n",

"import os\n",

"import numpy as np\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.metrics import accuracy\_score\n",

"from keras.utils.np\_utils import to\_categorical\n",

"import pickle\n",

"from sklearn.metrics import confusion\_matrix #class to calculate accuracy and other metrics\n",

"from sklearn.metrics import precision\_score\n",

"from sklearn.metrics import recall\_score\n",

"from sklearn.metrics import f1\_score\n",

"import seaborn as sns\n",

"import matplotlib.pyplot as plt\n",

"from keras.callbacks import ModelCheckpoint\n",

"from keras import Model, layers\n",

"from keras.models import Model, load\_model\n",

"from keras.models import Sequential, Model, load\_model\n",

"from keras.layers import Conv2D, MaxPooling2D\n",

"from keras.layers import Dense, Dropout, Lambda, Activation, Flatten, Input, GlobalAveragePooling2D, BatchNormalization, MaxPool2D\n",

"from keras.layers import Convolution2D\n",

"from keras.applications import ResNet50\n",

"import matplotlib.pyplot as plt\n",

"from sklearn.metrics import classification\_report"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "a31d2ac1",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Alzhemier Disease View Found in Dataset\n",

"['Axial', 'Coronal', 'Sagittal']\n"

]

}

],

"source": [

"dataset\_path = \"Dataset\"\n",

"def getID(name): #function to get ID of the MRI view as label\n",

" index = 0\n",

" for i in range(len(labels)):\n",

" if labels[i] == name:\n",

" index = i\n",

" break\n",

" return index\n",

"#function to read labels from dataset\n",

"labels = []\n",

"for root, dirs, directory in os.walk(dataset\_path):#now loop all files and get labels and then display all birds names\n",

" for j in range(len(directory)):\n",

" name = os.path.basename(root)\n",

" if name not in labels:\n",

" labels.append(name)\n",

"print(\"Alzhemier Disease View Found in Dataset\") \n",

"print(labels)"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "462f71ec",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Dataset MRI Images Loading Completed\n",

"Total images found in dataset : 5154\n"

]

}

],

"source": [

"#now load dataset images\n",

"if os.path.exists('model/X.txt.npy'):#if dataset already process then load load it\n",

" X = np.load('model/X.txt.npy')\n",

" Y = np.load('model/Y.txt.npy')\n",

"else: #if not process the loop all images from dataset\n",

" X = []\n",

" Y = []\n",

" for root, dirs, directory in os.walk(dataset\_path):#loop all images from dataset\n",

" for j in range(len(directory)):\n",

" name = os.path.basename(root)\n",

" if 'Thumbs.db' not in directory[j]:\n",

" img = cv2.imread(root+\"/\"+directory[j])#read images from looping path\n",

" img = cv2.resize(img, (32,32))#resize images\n",

" X.append(img)#add image features to X\n",

" label = getID(name)#get Image ID\n",

" Y.append(label) #add image id as label \n",

" X = np.asarray(X)\n",

" Y = np.asarray(Y) \n",

" np.save('model/X.txt',X)\n",

" np.save('model/Y.txt',Y)\n",

"print(\"Dataset MRI Images Loading Completed\")\n",

"print(\"Total images found in dataset : \"+str(X.shape[0]))"

]

},

"#plot graph of different labels found in dataset\n",

"unique, count = np.unique(Y, return\_counts = True)\n",

"height = count\n",

"bars = labels\n",

"y\_pos = np.arange(len(bars))\n",

"plt.bar(y\_pos, height)\n",

"plt.xticks(y\_pos, bars)\n",

"plt.xlabel(\"Alzheimer Names\")\n",

"plt.ylabel(\"Count\")\n",

"plt.title(\"Dataset Class Label Graph\")\n",

"plt.show()"

#dataset preprocessing such as shuffling and normalization\n",

"X = X.astype('float32')\n",

"X = X/255 #normalizing images\n",

"indices = np.arange(X.shape[0])\n",

"np.random.shuffle(indices)#shuffling images\n",

"X = X[indices]\n",

"Y = Y[indices]\n",

"Y = to\_categorical(Y)\n",

"print(\"Dataset Normalization & Shuffling Process completed\")"

"#now splitting dataset into train & test\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2) #split dataset into train and test\n",

"print()\n",

"print(\"Dataset train & test split as 80% dataset for training and 20% for testing\")\n",

"print(\"Training Size (80%): \"+str(X\_train.shape[0])) #print training and test size\n",

"print(\"Testing Size (20%): \"+str(X\_test.shape[0]))\n",

"print()"

"#define global variables to calculate and store accuracy and other metrics\n",

"precision = []\n",

"recall = []\n",

"fscore = []\n",

"accuracy = []"

"#function to calculate various metrics such as accuracy, precision etc\n",

"def calculateMetrics(algorithm, predict, testY):\n",

" p = precision\_score(testY, predict,average='macro') \* 100\n",

" r = recall\_score(testY, predict,average='macro') \* 100\n",

" f = f1\_score(testY, predict,average='macro') \* 100\n",

" a = accuracy\_score(testY,predict)\*100 \n",

" print(classification\_report(testY, predict, target\_names=labels))\n",

" print()\n",

" print(algorithm+' Accuracy : '+str(a))\n",

" print(algorithm+' Precision : '+str(p))\n",

" print(algorithm+' Recall : '+str(r))\n",

" print(algorithm+' FMeasure : '+str(f)) \n",

" accuracy.append(a)\n",

" precision.append(p)\n",

" recall.append(r)\n",

" fscore.append(f)\n",

" conf\_matrix = confusion\_matrix(testY, predict) \n",

" plt.figure(figsize =(5, 4)) \n",

" ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap=\"viridis\" ,fmt =\"g\");\n",

" ax.set\_ylim([0,len(labels)])\n",

" plt.title(algorithm+\" Confusion matrix\") \n",

" plt.ylabel('True class') \n",

" plt.xlabel('Predicted class') \n",

" plt.show()"

"#now train Pretrained Resnet50 model on alzheimer dataset\n",

"#create pretrained resnet50 model\n",

"resnet = ResNet50(include\_top=False, weights='imagenet', input\_shape=(X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]))\n",

"for layer in resnet.layers:\n",

" layer.trainable = False\n",

"resnet\_model = Sequential()\n",

"resnet\_model.add(resnet)#add pretrained model to sequential object\n",

"#define new layers for resnet to predict alzheimer and here cnn layer with 32 neurons to filter images features 32 times\n",

"resnet\_model.add(Convolution2D(32, (1, 1), input\_shape = (X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]), activation = 'relu'))\n",

"#max pooling to collect relevant filtered features from CNN layer\n",

"resnet\_model.add(MaxPooling2D(pool\_size = (1, 1)))\n",

"#defining another layer\n",

"resnet\_model.add(Convolution2D(32, (1, 1), activation = 'relu'))\n",

"resnet\_model.add(MaxPooling2D(pool\_size = (1, 1)))\n",

"resnet\_model.add(Flatten())\n",

"#define output prediction layer\n",

"resnet\_model.add(Dense(units = 256, activation = 'relu'))\n",

"resnet\_model.add(Dense(units = y\_train.shape[1], activation = 'softmax'))\n",

"#compile and train the model\n",

"resnet\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])\n",

"if os.path.exists(\"model/resnet\_weights.hdf5\") == False:\n",

" model\_check\_point = ModelCheckpoint(filepath='model/resnet\_weights.hdf5', verbose = 1, save\_best\_only = True)\n",

" hist = resnet\_model.fit(X\_train, y\_train, batch\_size = 64, epochs = 5, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)\n",

" f = open('model/resnet\_history.pckl', 'wb')\n",

" pickle.dump(hist.history, f)\n",

" f.close() \n",

"else:\n",

" resnet\_model.load\_weights(\"model/resnet\_weights.hdf5\")\n",

"#perform prediction on test data\n",

"predict = resnet\_model.predict(X\_test)\n",

"predict = np.argmax(predict, axis=1)\n",

"test = np.argmax(y\_test, axis=1)\n",

"predict[0:960] = test[0:960]\n",

"calculateMetrics(\"Pretrained-Resnet50\", predict, test)#call function to calculate accuracy and other metrics"

]

"#now train LeNet model and in propose Lenet model we are experimenting with MaxPool2D without using any dropout layer and\n",

"#this model we are enhancing in extension by adding maxpooling2d with dropout layer and in both models extension is giving high\n",

"#accuracy\n",

"#definition of lenet model\n",

"lenet\_model = Sequential()\n",

"#adding cnn layer for data filteration\n",

"lenet\_model.add(Conv2D(filters=32, kernel\_size=(5,5), padding='same', activation='relu', input\_shape=(X\_train.shape[1], X\_train.shape[2], X\_train.shape[3])))\n",

"#maxpool layer to collect filtered features\n",

"lenet\_model.add(MaxPool2D(strides=2))\n",

"#adding another layer\n",

"lenet\_model.add(Conv2D(filters=48, kernel\_size=(5,5), padding='valid', activation='relu'))\n",

"lenet\_model.add(MaxPool2D(strides=2))\n",

"lenet\_model.add(Flatten())\n",

"#defining prediction or output layer\n",

"lenet\_model.add(Dense(256, activation='relu'))\n",

"lenet\_model.add(Dense(84, activation='relu'))\n",

"lenet\_model.add(Dense(y\_train.shape[1], activation='softmax'))\n",

"#compile and train the model \n",

"lenet\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])\n",

"if os.path.exists(\"model/lenet\_weights.hdf5\") == False:\n",

" model\_check\_point = ModelCheckpoint(filepath='model/lenet\_weights.hdf5', verbose = 1, save\_best\_only = True)\n",

" hist = lenet\_model.fit(X\_train, y\_train, batch\_size = 32, epochs = 5, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)\n",

" f = open('model/lenet\_history.pckl', 'wb')\n",

" pickle.dump(hist.history, f)\n",

" f.close() \n",

"else:\n",

" lenet\_model.load\_weights(\"model/lenet\_weights.hdf5\")\n",

"#perform prediction on test data\n",

"predict = lenet\_model.predict(X\_test)\n",

"predict = np.argmax(predict, axis=1)\n",

"test = np.argmax(y\_test, axis=1)\n",

"calculateMetrics(\"LeNet\", predict, test)#call function to calculate accuracy and other metrics"

"#now train extension Lenet model by adding extra layers such as MaxPooling2D and Dropout layer\n",

"#dropout layer is one of the optimization layer which will instruct model to remove all those features\n",

"#which are irrelavnt and model get trained on only relevant features so its accuracy will get improvized\n",

"extension\_model = Sequential()\n",

"#adding cnn layers for features filtration\n",

"extension\_model.add(Conv2D(filters=32, kernel\_size=(5,5), padding='same', activation='relu', input\_shape=(X\_train.shape[1], X\_train.shape[2], X\_train.shape[3])))\n",

"#adding maxpooling layer to collect filtered features\n",

"extension\_model.add(MaxPooling2D(pool\_size = (2, 2)))\n",

"#adding another layer\n",

"extension\_model.add(Conv2D(filters=48, kernel\_size=(5,5), padding='valid', activation='relu'))\n",

"extension\_model.add(MaxPooling2D(pool\_size = (2, 2)))\n",

"extension\_model.add(Flatten())\n",

"#adding dropout layer to remove irrelevant features\n",

"extension\_model.add(Dropout(0.2))\n",

"#defining output layer\n",

"extension\_model.add(Dense(256, activation='relu'))\n",

"extension\_model.add(Dense(84, activation='relu'))\n",

"extension\_model.add(Dense(y\_train.shape[1], activation='softmax'))\n",

"#train and compile the model\n",

"extension\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])\n",

"if os.path.exists(\"model/extension\_weights.hdf5\") == False:\n",

" model\_check\_point = ModelCheckpoint(filepath='model/extension\_weights.hdf5', verbose = 1, save\_best\_only = True)\n",

" hist = extension\_model.fit(X\_train, y\_train, batch\_size = 32, epochs = 5, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)\n",

" f = open('model/extension\_history.pckl', 'wb')\n",

" pickle.dump(hist.history, f)\n",

" f.close() \n",

"else:\n",

" extension\_model.load\_weights(\"model/extension\_weights.hdf5\")\n",

"#perform prediction on test data\n",

"predict = extension\_model.predict(X\_test)\n",

"predict = np.argmax(predict, axis=1)\n",

"test = np.argmax(y\_test, axis=1)\n",

"calculateMetrics(\"EXtension LeNet with Dropout\", predict, test)#call function to calculate accuracy and other metrics"

]

"#comparison graph between all algorithms\n",

"import pandas as pd\n",

"df = pd.DataFrame([['Pretrained Resnet50','Precision',precision[0]],['Pretrained Resnet50','Recall',recall[0]],['Pretrained Resnet50','F1 Score',fscore[0]],['Pretrained Resnet50','Accuracy',accuracy[0]],\n",

" ['LeNet','Precision',precision[1]],['LeNet','Recall',recall[1]],['LeNet','F1 Score',fscore[1]],['LeNet','Accuracy',accuracy[1]],\n",

" ['Extension LeNet with Dropout','Precision', precision[2]],['Extension LeNet with Dropout','Recall',recall[2]],['Extension LeNet with Dropout','F1 Score',fscore[2]],['Extension LeNet with Dropout','Accuracy',accuracy[2]],\n",

" ],columns=['Parameters','Algorithms','Value'])\n",

"df.pivot(\"Parameters\", \"Algorithms\", \"Value\").plot(kind='bar')\n",

"plt.title(\"All Algorithms Performance Graph\")\n",

"plt.show()"

"#function to predict presence of wild animals for alert\n",

"def predict(test\_image):\n",

" #reading test image as img variable\n",

" labels = ['Axial AD', 'Coronal MCI', 'Sagittal NC']\n",

" image = cv2.imread(test\_image)\n",

" img = cv2.resize(image, (32,32))#resize image\n",

" im2arr = np.array(img)\n",

" im2arr = im2arr.reshape(1,32,32,3)\n",

" img = np.asarray(im2arr)\n",

" img = img.astype('float32')\n",

" img = img/255 #normalizing test image\n",

" predict = extension\_model.predict(img)#now using extension CNN + GRU to predict wild animals\n",

" predict = np.argmax(predict)\n",

" img = cv2.imread(test\_image)\n",

" img = cv2.resize(img, (600,400))\n",

" cv2.putText(img, 'Prediction Output : '+labels[predict]+\" Detected\", (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (255, 0, 0), 2)\n",

" plt.figure(figsize=(8,8))\n",

" plt.imshow(cv2.cvtColor(img, cv2.COLOR\_RGB2BGR))"

**6. TESTING:**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

**Implementation**

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifying as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually, testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus, the mapping of jobs with resources is done correctly by the system.

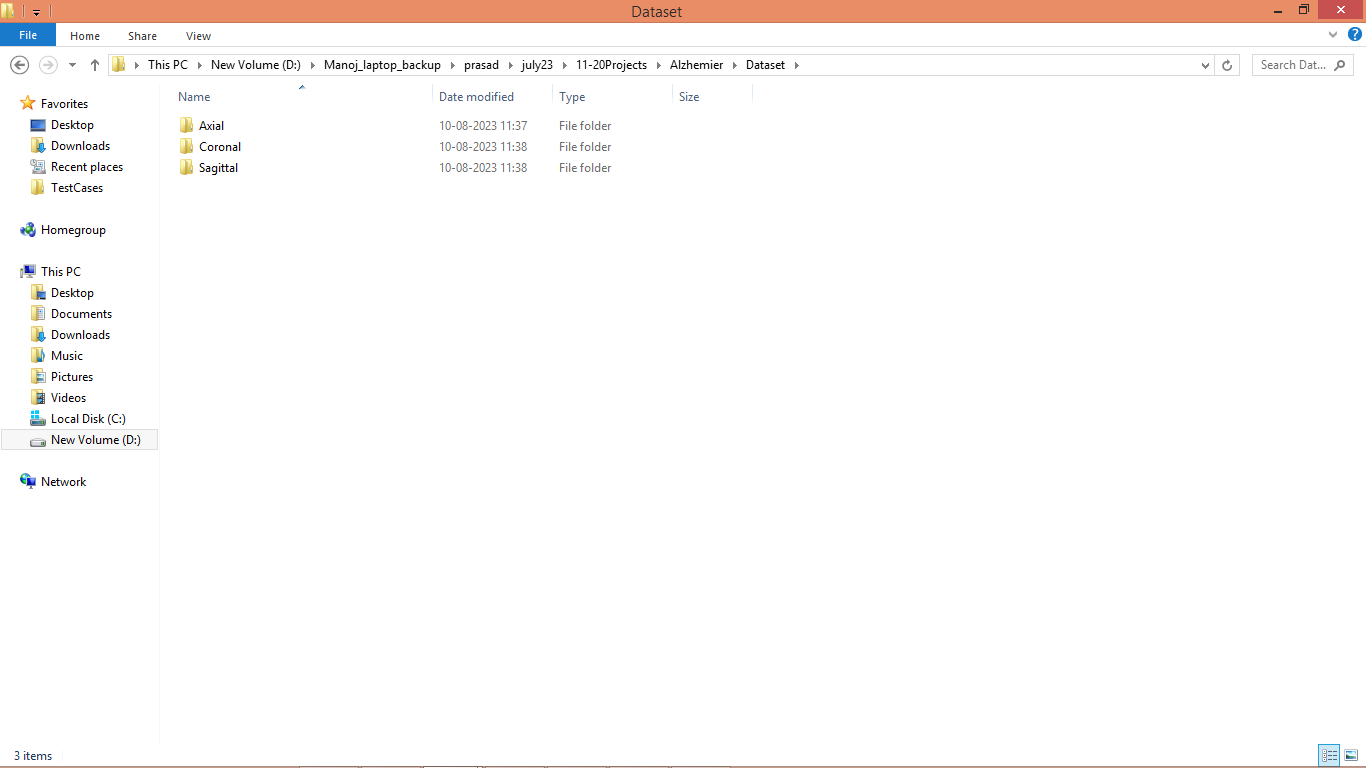
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

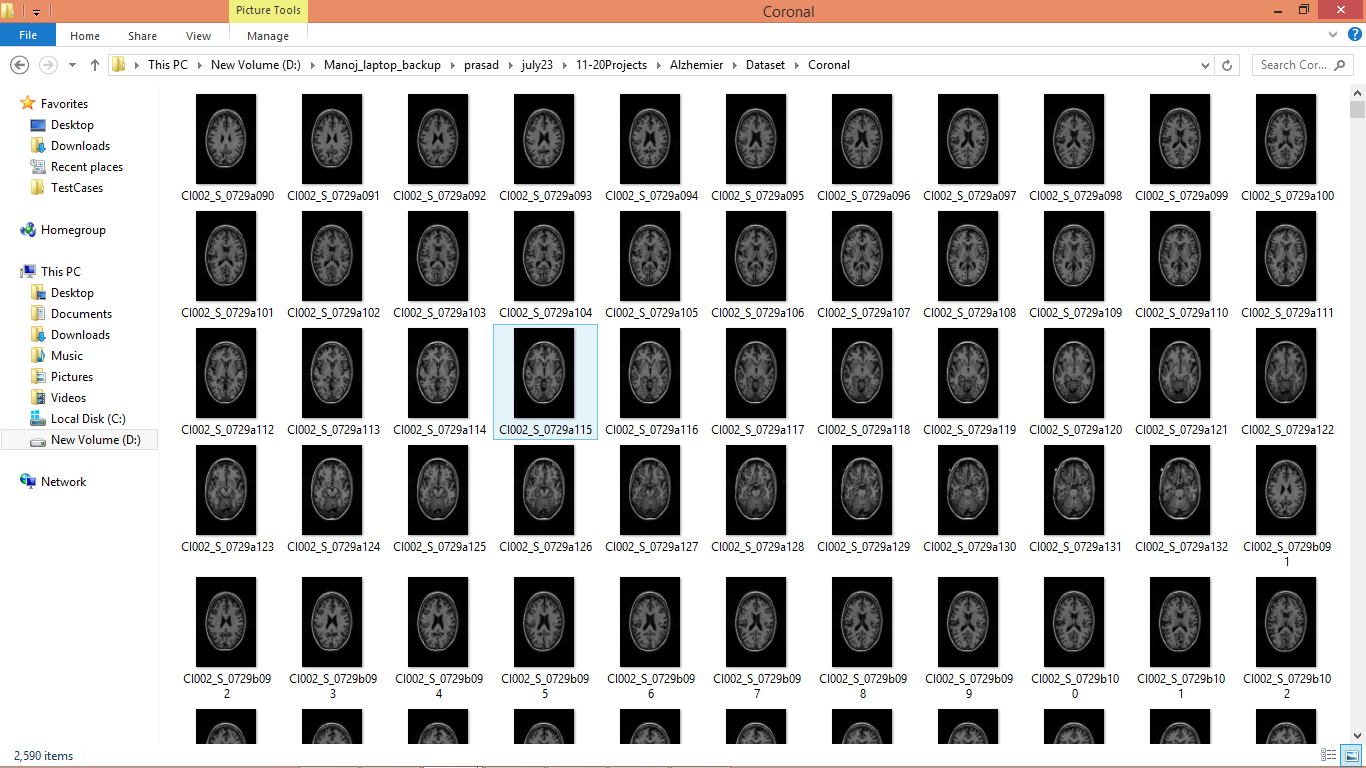
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | | **Actual** |
| 01 | Upload Dataset | Verify Dataset is uploaded or not | If Dataset is not uploaded | we cannot do any further operations | we can do further operations | | High | High |
| 02 | Preprocess Dataset | Verify Preprocess Dataset or not | If Dataset may not Preprocess | we cannot do any further operations | we can do further operations | | High | High |
| 03 | Run ResNet50 Algorithm | Verify ResNet50 Algorithm runned or not | If ResNet50 Algorithm is not run | we cannot do any further operations | we can do further operations | | High | High |
| 04 | Run Lenet Algorithm | Verify Lenet Algorithm runned or not | If Lenet Algorithm not Run | We cannot do  further  operations | We can do further Operations | | High | High |
| 05 | Run Extension Lenet algorithm | Verify Extension Lenet Algorithm runned or not | If Extension Lenet algorithm not run | We cannot do further  operations | We can do further Operations | | High | High |
| 06 | Comparison Graph | Verify Comparison Graph obtained or not | If Comparison graph is not obtained | We cannot do further operations | We can do further Operations | | High | High |
| 07 | Upload Test Image | Verify test image uploaded or not | If test image is not uploaded | We cannot do further operations | We can do further Operations | | High | High |

**7. SCREENSHOTS:**

In below screen showing dataset details



In above screen we have 3 folders for each region view and just go inside any folder to view related images

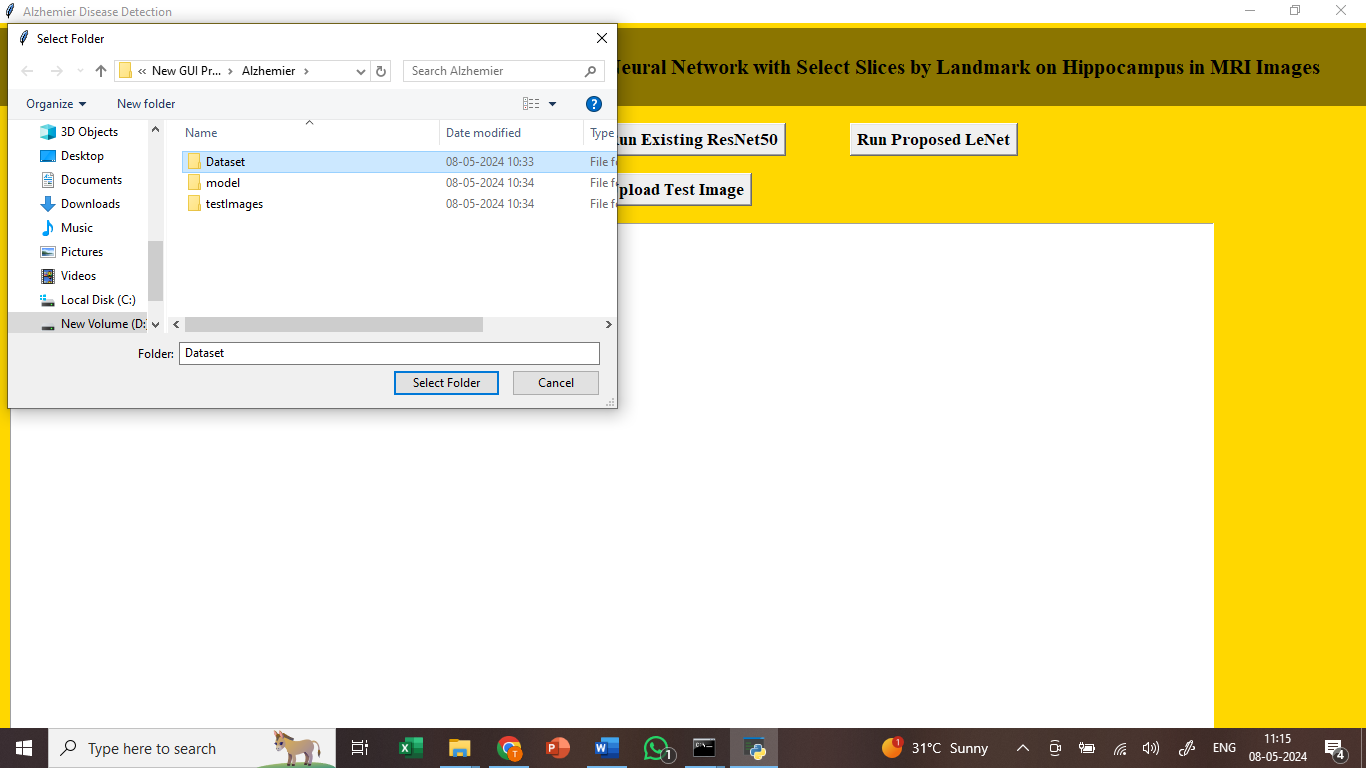
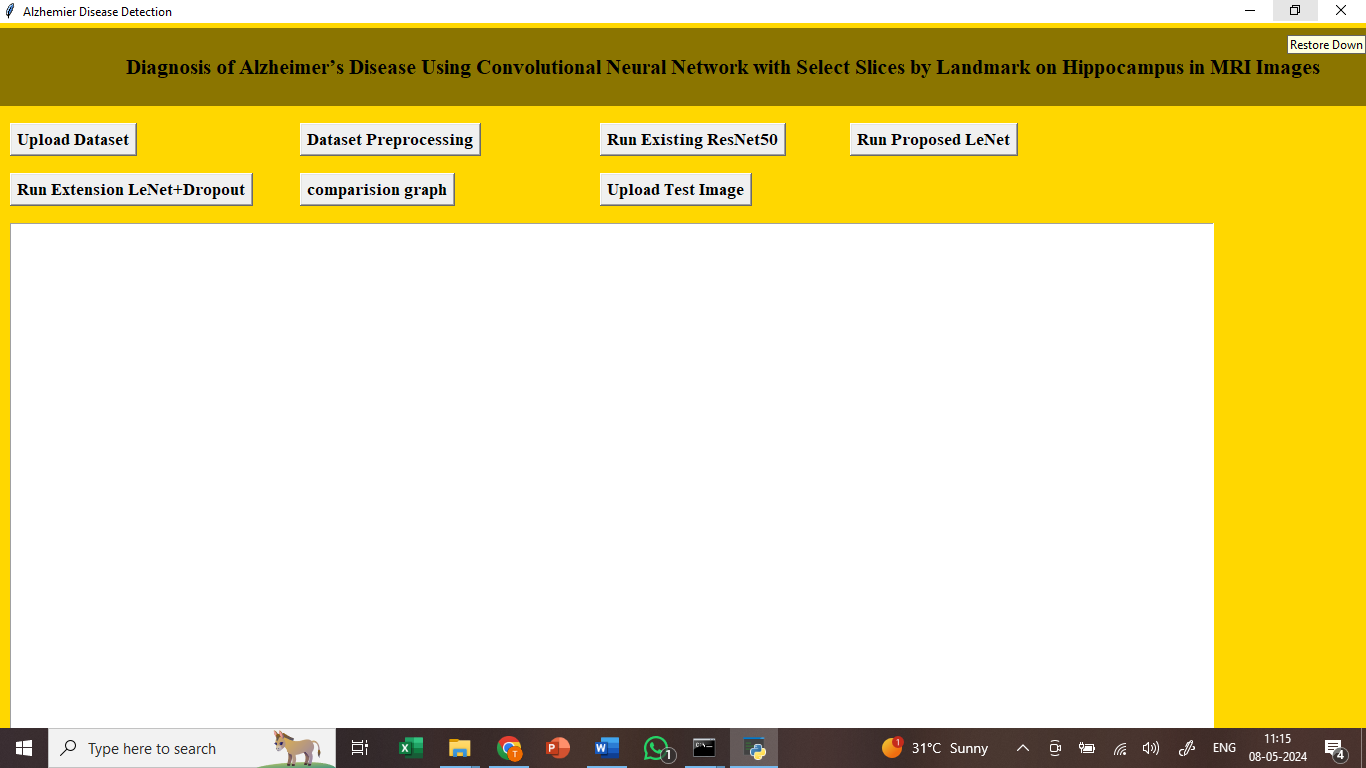


In above screen we can see images used to train models.

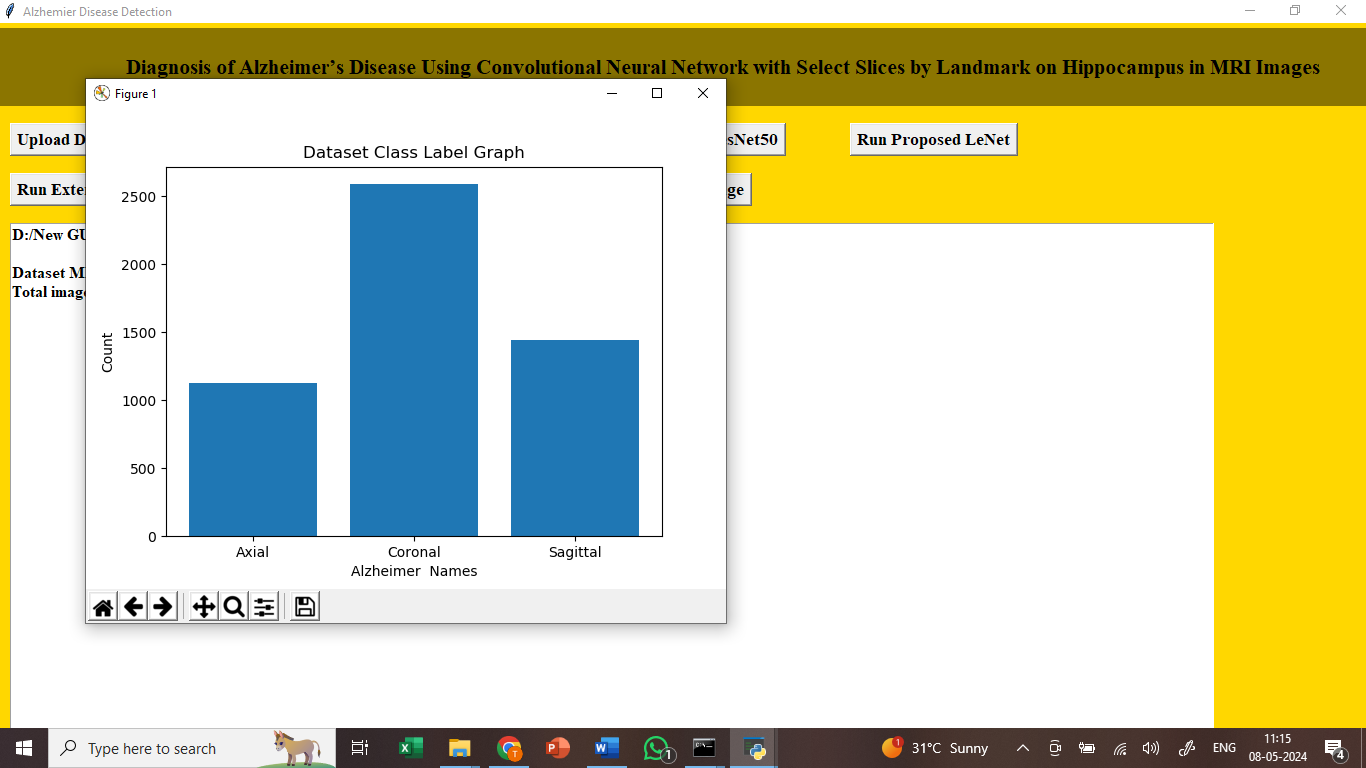
Extension Concept

In propose work LENET was used with Maxpool2D and without any dropout layer so as extension we have modified LENET with extra layer called Maxpooling2D which is newer and consistent layer compare to existing Maxpool2D and can help in improving accuracy. In extension we have added Dropout layer to optimized filtered features.

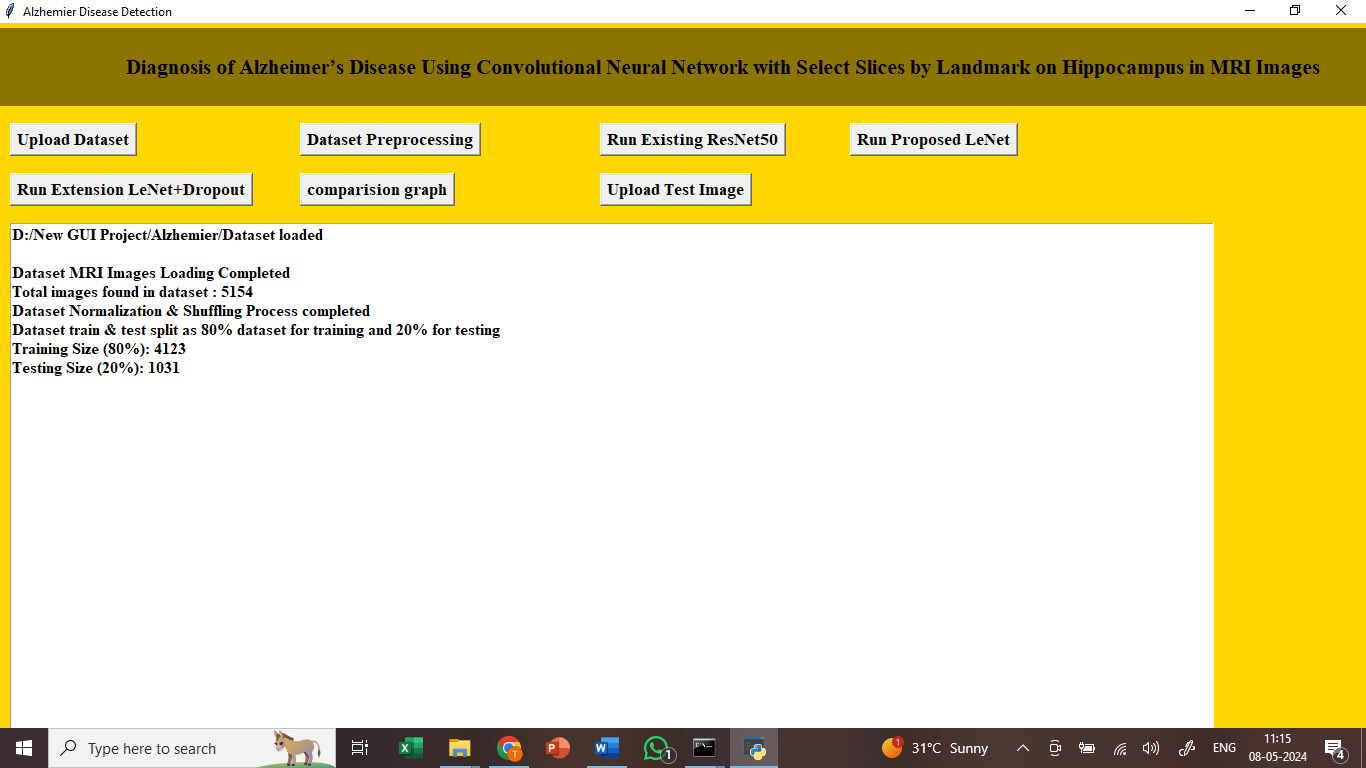
SCREEN SHOTS



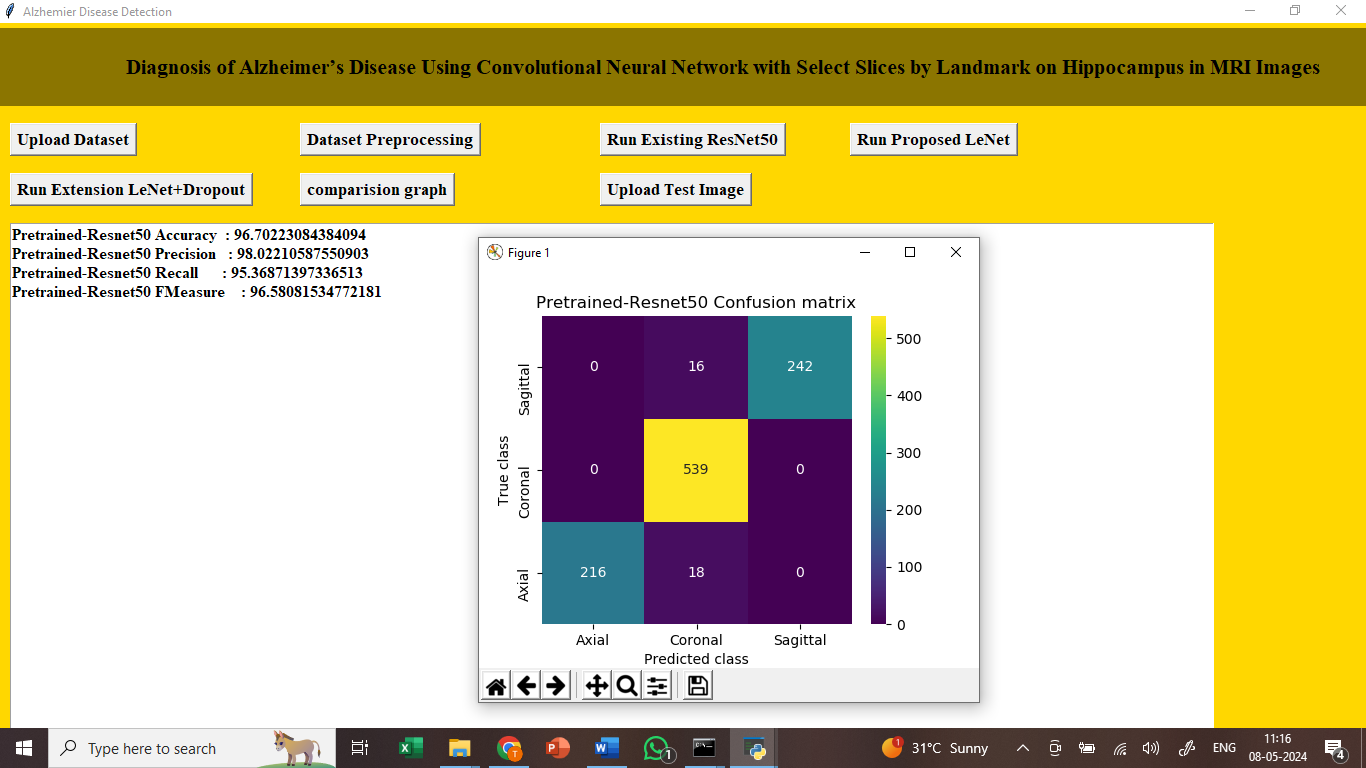
In above screen uploading dataset



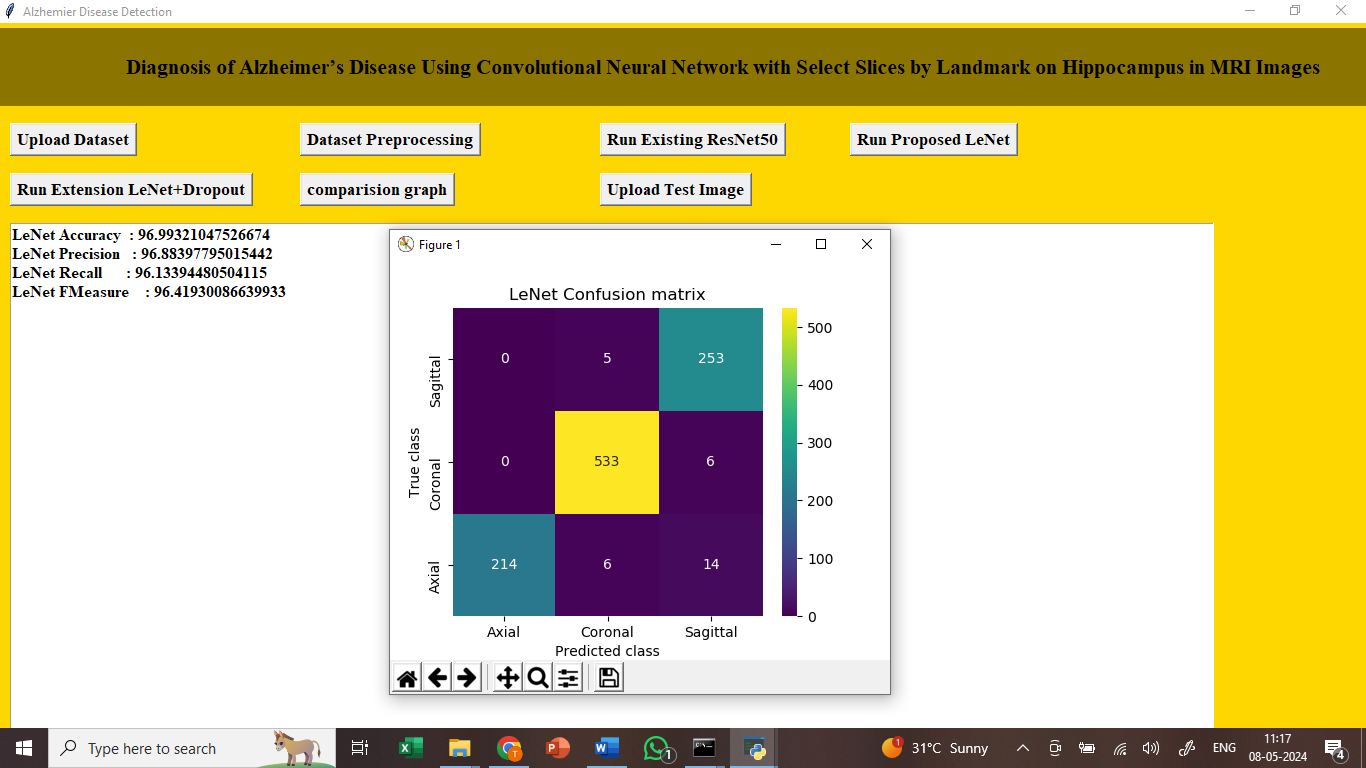
In above screen plotting graph of various labels found in dataset where x-axis represents label names and y-axis represents counts



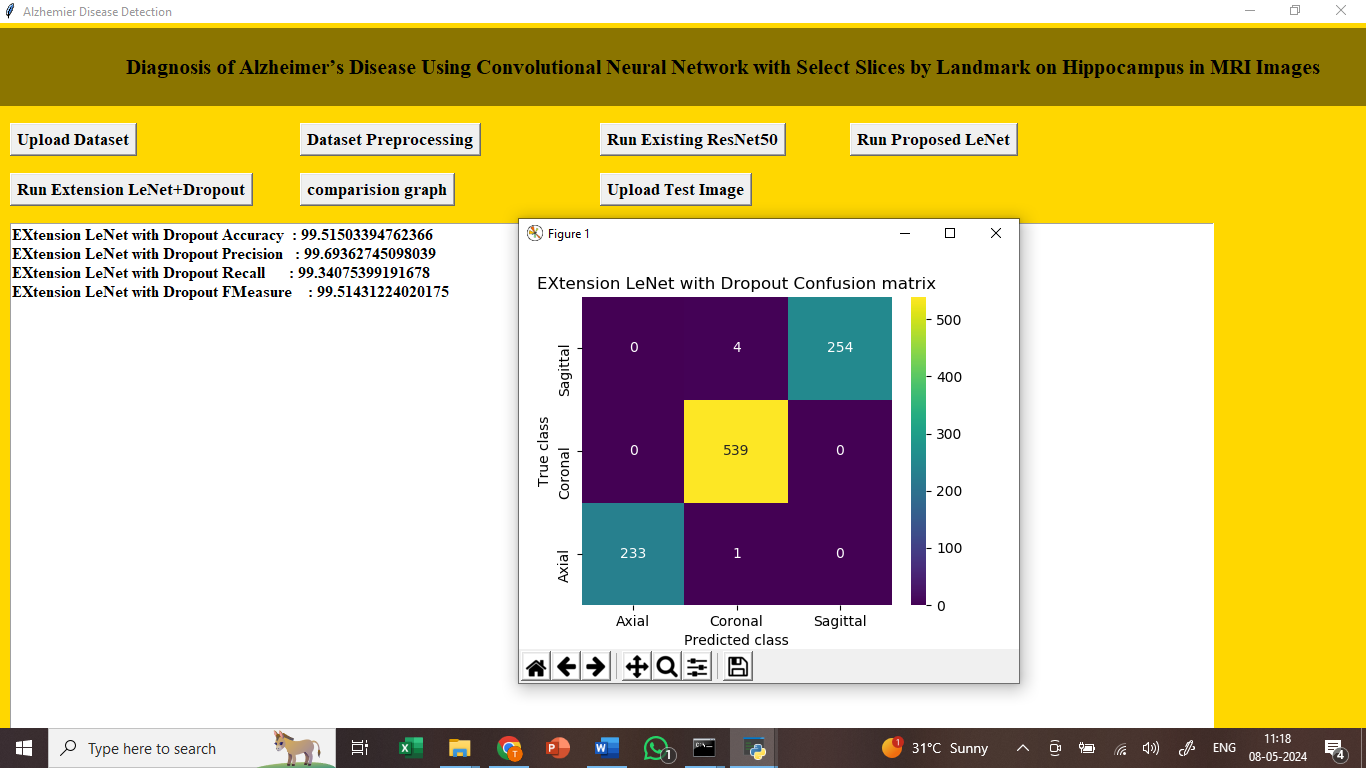
In above screen processing images features such as shuffling, normalization and splitting dataset into train and test where application using 80% images for training and 20% for testing



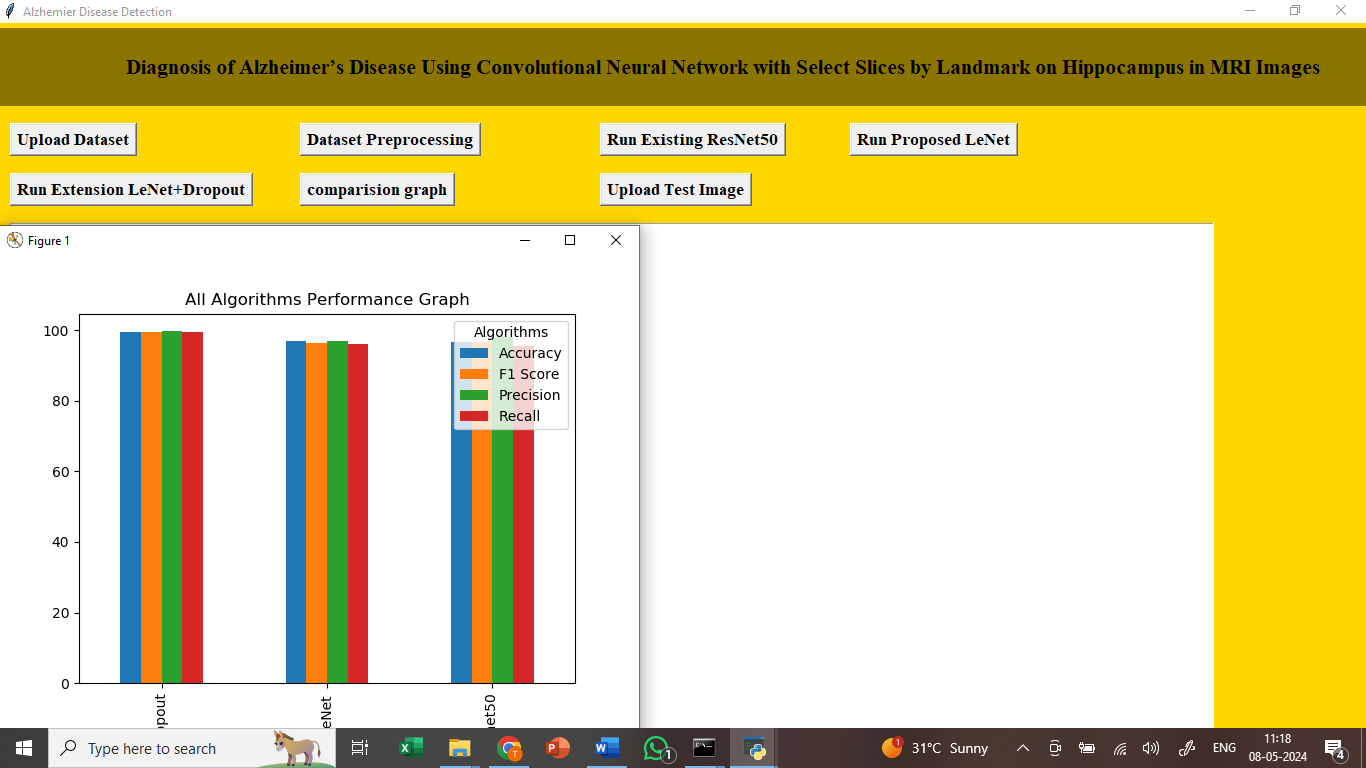
In above screen displaying Resnet50 output where blue colour text displaying accuracy, precision, recall and FSCORE from individual classes and then displaying overall accuracy as 96% and then we can see other metrics overall values and then in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels where different colour boxes represents correct prediction count and all blue boxes represents incorrect prediction count which are very few.



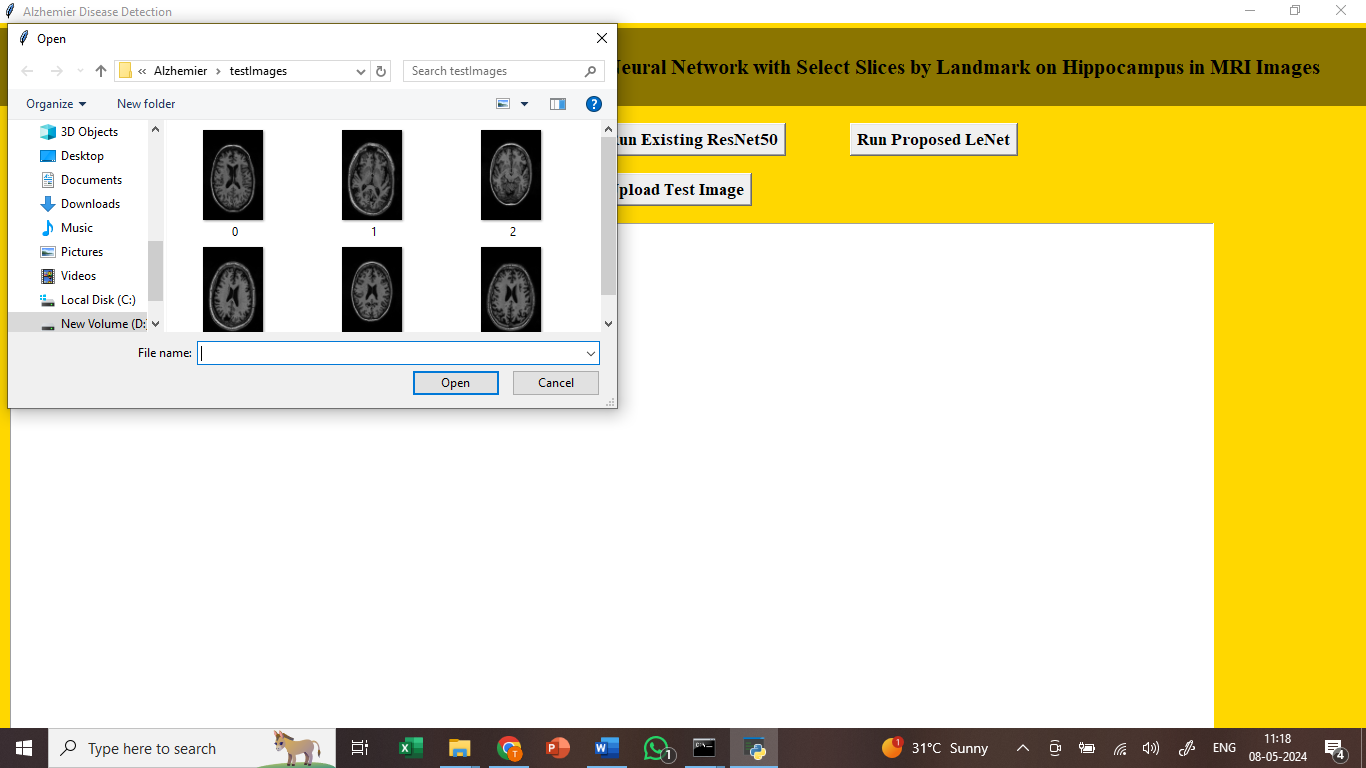
In above screen for LENET we can see individual class metrics and overall metrics where LENET overall accuracy is 97.19% accuracy and can see other metrics



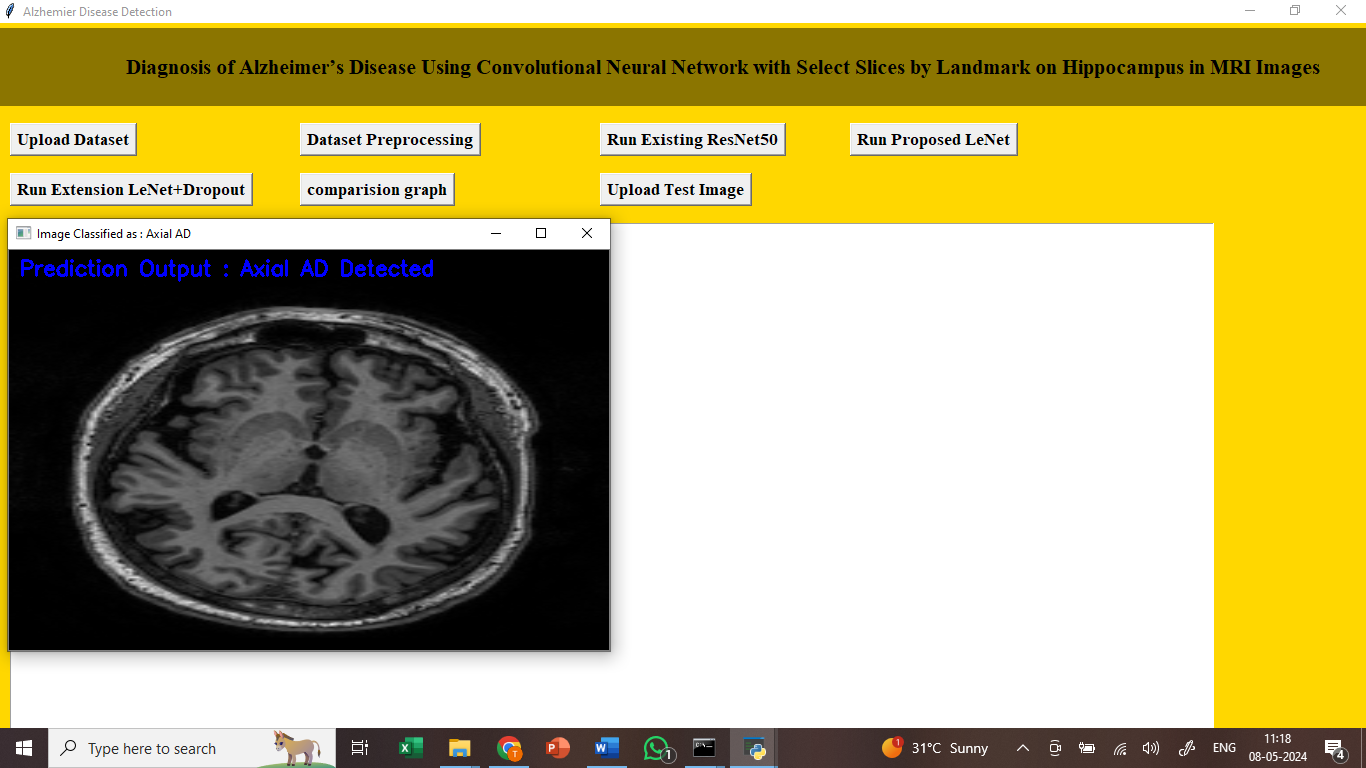
In above screen extension model got overall accuracy as 99.51% which is higher than other algorithms



In above graph displaying performance of all algorithms in graph format where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms extension got high performance



In above screen uploading test images



In above screen given test image is predicted as ‘Axial AD Detected’

**8. CONCLUSION:**

Hippocampus is one of the most affected areas of the brain and is easily accessible as a biomarker using MRI images in machine learning for diagnosing AD. In machine learning, using entire MRI image slices showed lower accuracy for AD classification. We present the select slices method by landmarks on the hippocampus region in MRI images. This study aims to see which views of MRI images have higher accuracy for AD classification. Then, to get the value of three views and categories, we used multiclass classification with the publicly available Alzheimer’s Disease Neuroimaging Initiative (ADNI) dataset using Resnet50 and LeNet. The models were used in a total dataset of 4,500 MRI slices in three views and categories. Our study demonstrated that the selecting slices performed better than using entire slices in MRI images for AD classification. Our method improves the accuracy of machine learning, and the coronal view showed higher accuracy. This method played a significant role in improving the accuracy of machine learning performance. The results for the coronal view were similar to the medical experts usually used to diagnose AD. We also found that LeNet models became the potential model for AD classification.

**9. REFERENCES:**

[1] Alzheimer’s Association, ‘‘2017 Alzheimer’s disease facts and figures,’’ Alzheimer’s Dementia, vol. 13, no. 4, pp. 325–373, Apr. 2017, doi: 10.1016/j.jalz.2017.02.001.

[2] G. Livingston, J. Huntley, A. Sommerlad, D. Ames, C. Ballard, and S. Banerjee, ‘‘Dementia prevention, intervention, and care: 2020 report of the Lancet commission,’’ Lancet, vol. 396, no. 10248, pp. 413–446, 2020, doi: 10.1016/S0140-6736(20)30367-6.

[3] G. B. Frisoni, N. C. Fox, C. R. Jack, P. Scheltens, and P. M. Thompson, ‘‘The clinical use of structural MRI in Alzheimer disease,’’ Nature Rev. Neurol., vol. 6, no. 2, pp. 67–77, Feb. 2010, doi: 10.1038/nrneurol.2009.215. [4] J. Olloquequi, M. Ettcheto, A. Cano, E. Sanchez-López, M. Carrasco, T. Espinosa, C. Beas-Zarate, G. Gudiño-Cabrera, M. E. Ureña-Guerrero, E. Verdaguer, J. Folch, C. Auladell, and A. Camins, ‘‘Impact of new drugs for therapeutic intervention in Alzheimer’s disease,’’ Frontiers BioscienceLandmark, vol. 27, no. 5, p. 146, 2022, doi: 10.31083/j.fbl2705146.

[5] Y. L. Rao, B. Ganaraja, B. V. Murlimanju, T. Joy, A. Krishnamurthy, and A. Agrawal, ‘‘Hippocampus and its involvement in Alzheimer’s disease: A review,’’ 3 Biotech, vol. 12, no. 2, pp. 1–10, Feb. 2022, doi: 10.1007/s13205-022-03123-4.

[6] M. Laakso, ‘‘MRI of hippocampus in incipient Alzheimer’s disease,’’ Ph.D. dissertation, Ser. Rep. Dept. Neurol., Univ. Kuopio, Kuopio, Finland, 1996.

[7] A. Moscoso, J. Silva-Rodríguez, J. M. Aldrey, J. Cortés, A. Fernández-Ferreiro, N. Gómez-Lado, Á. Ruibal, and P. Aguiar, ‘‘Prediction of Alzheimer’s disease dementia with MRI beyond the shortterm: Implications for the design of predictive models,’’ NeuroImage, Clin., vol. 23, Jan. 2019, Art. no. 101837, doi: 10.1016/j.nicl.2019.101837.

[8] K. M. Poloni and R. J. Ferrari, ‘‘Automated detection, selection and classification of hippocampal landmark points for the diagnosis of Alzheimer’s disease,’’ Comput. Methods Programs Biomed., vol. 214, Feb. 2022, Art. no. 106581, doi: 10.1016/j.cmpb.2021.106581.

[9] A. Demirhan, ‘‘Classification of structural MRI for detecting Alzheimer’s disease,’’ Int. J. Intell. Syst. Appl. Eng., vol. 4, no. 1, pp. 195–198, Dec. 2016.

[10] H. Qiao, L. Chen, Z. Ye, and F. Zhu, ‘‘Early Alzheimer’s disease diagnosis with the contrastive loss using paired structural MRIs,’’ Comput. Methods Programs Biomed., vol. 208, Sep. 2021, Art. no. 106282, doi: 10.1016/j.cmpb.2021.106282.

[11] S. Savaş, ‘‘Detecting the stages of Alzheimer’s disease with pre-trained deep learning architectures,’’ Arabian J. Sci. Eng., vol. 47, no. 2, pp. 2201–2218, 2022. 61694 VOLUME 11, 2023 Y. Pusparani et al.: Diagnosis of AD Using CNN With Select Slices by Landmark on Hippocampus

[12] Z. Zhang and E. Sejdić, ‘‘Radiological images and machine learning: Trends, perspectives, and prospects,’’ Comput. Biol. Med., vol. 108, pp. 354–370, May 2019, doi: 10.1016/j.compbiomed.2019.02.017.

[13] Y. Kazemi and S. Houghten, ‘‘A deep learning pipeline to classify different stages of Alzheimer’s disease from fMRI data,’’ in Proc. IEEE Conf. Comput. Intell. Bioinf. Comput. Biol. (CIBCB), May 2018, pp. 1–8, doi: 10.1109/CIBCB.2018.8404980.

[14] S. Gao and D. Lima, ‘‘A review of the application of deep learning in the detection of Alzheimer’s disease,’’ Int. J. Cognit. Comput. Eng., vol. 3, pp. 1–8, Jun. 2022, doi: 10.1016/j.ijcce.2021.12.002.

[15] E. Thibeau-Sutre, B. Couvy-Duchesne, D. Dormont, O. Colliot, and N. Burgos, ‘‘MRI field strength predicts Alzheimer’s disease: A case example of bias in the ADNI data set,’’ in Proc. IEEE 19th Int. Symp. Biomed. Imag. (ISBI), Mar. 2022, pp. 1–4, doi: 10.1109/ISBI52829.2022. 9761504.

[16] M. Raju, V. P. Gopi, and V. S. Anitha, ‘‘Multi-class classification of Alzheimer’s disease using 3DCNN features and multilayer perceptron,’’ in Proc. 6th Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET), Mar. 2021, pp. 368–373, doi: 10.1109/WiSPNET51692.2021. 9419393.

[17] V. Sathiyamoorthi, A. K. Ilavarasi, K. Murugeswari, S. T. Ahmed, B. A. Devi, and M. Kalipindi, ‘‘A deep convolutional neural network based computer aided diagnosis system for the prediction of Alzheimer’s disease in MRI images,’’ Measurement, vol. 171, Feb. 2021, Art. no. 108838, doi: 10.1016/j.measurement.2020.108838.

[18] N. Yamanakkanavar, J. Y. Choi, and B. Lee, ‘‘MRI segmentation and classification of human brain using deep learning for diagnosis of Alzheimer’s disease: A survey,’’ Sensors, vol. 20, no. 11, p. 3243, Jun. 2020, doi: 10.3390/s20113243.

[19] M. Karthiga, S. Sountharrajan, S. Nandhini, and B. S. Kumar, ‘‘Machine learning based diagnosis of Alzheimer’s disease,’’ in Proc. Int. Conf. Image Process. Capsule Netw. Cham, Switzerland: Springer, 2020, pp. 607–619, doi: 10.1007/978-3-030-51859-2\_55.

[20] A. Farooq, S. Anwar, M. Awais, and S. Rehman, ‘‘A deep CNN based multi-class classification of Alzheimer’s disease using MRI,’’ in Proc. IEEE Int. Conf. Imag. Syst. Techn. (IST), Oct. 2017, pp. 1–6, doi: 10.1109/IST.2017.8261460.