A

Major Project

On

PREDICTION AND CLASSIFICATION OF ALZHEIMER DISEASE SEVERITY USING DEEP LEARNING MODEL

(Submitted in partial fulfillment of the requirements for the award of degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

BY

Akshitha Sabbineni(187R1A05C7)

A Sivasankar(187R1A05C3)

Ajmeera Madhu(187R1A05C5)

Under the Guidance of

A Kiran Kumar

(Assistant professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**PREDICTION AND CLASSIFICATION OF ALZHEIMER DISEASE SEVERITY USING DEEP LEARNING MODEL**" being submitted by **A SIVASANKAR (187R1A05C3), AKSHITHA SABBINENI (187R1A05C7), AJMEERA MADHU (187R1A05C5)** partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mr. A KIRAN KUMAR
Assistant Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. A. Kiran Kumar**, Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to **Sri. Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

AKSHITHA SABBINENI(187R1A05C7)

A SIVASANKAR(187R1A05C3)

AJMEERA MADHU(187R1A05C5)

ABSTRACT

Alzheimer's disease is a progressive neurologic disorder that causes the brain to shrink (atrophy) and brain cells to die. Alzheimer's disease is the most common cause of dementia a continuous decline in thinking, behavioral and social skills that affects a person's ability to function independently. Approximately 5.8 million people in the United States age 65 and older live with Alzheimer's disease. Of those, 80% are 75 years old and older. Out of the approximately 50 million people worldwide with dementia, between 60% and 70% are estimated to have Alzheimer's disease. The early signs of the disease include forgetting recent events or conversations. As the disease progresses, a person with Alzheimer's disease will develop severe memory impairment and lose the ability to carry out everyday tasks. Medications may temporarily improve or slow progression of symptoms. These treatments can sometimes help people with Alzheimer's disease maximize function and maintain independence for a time. Different programs and services can help support people with Alzheimer's disease and their caregivers. There is no treatment that cures Alzheimer's disease or alters the disease process in the brain. In advanced stages of the disease, complications from severe loss of brain function — such as dehydration, malnutrition or infection result in death

LIST OF FIGURES / TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENT

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

This project is titled as "Prediction and Classification of Alzheimer Disease Severity Using Deep Learning Model". This machine learning model helps to detect early alzheimers using MRI(magnetic resonance imaging) scans of brain. The machine learning model uses CNN(convolutional neural network) algorithm to classify the MRI scans into 4 classes namely Mild demented, Moderate demented, non demented, very mild demented.

1.2 PROJECT PURPOSE

This model helps doctors in the hospital to detect the alzheimers as early as possible. Alzheimers cannot be cured if it is severely affected if it is identified in early stages then it will be easy for the therapy. This model not only helps in detecting alzheimers, it also helps in identifying other brain oriented diseases.

1.3 PROJECT FEATURES

Plotting of various graphs which help analyse the data completely and have good understanding regarding the data. Understanding the current patterns. The Alzheimer_s dataset consists of four classes inside which it consists of training and testing data which are preprocessed so that the machine learning model can understand the image.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.Analysis specifies what the system should do.

2.1 PROBLEM DEFINITION

There is no cure for Alzheimer's, but there are treatments that can delay the onset of the disease. Early detection is critical, and the ability to diagnose the disease early often means better outcomes for those who have it. This dataset includes images taken from all over the web of brains that have either no indications of Alzheimer's or indications of very mild, mild, or moderate Alzheimer's. Using an array of techniques, we can create a convolutional neural network (CNN) using Keras that is capable of making predictions about similar pictures of the brain.

2.2 EXISTING SYSTEM

Escudero et al. proposed a ML approach using biomarkers in their paper. They tested a personalized classifier for the disease using a method learning locally weighted and biomarkers. The methodology attempts to classify the subject first and then later decides which biomarker to order. They classified the patients who were MCI who advanced to AD inside a year against the individuals who didn't.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Such types can be vigorously standardized in human vision but need significant advances in figuring out how to dodge perils from overlooked and underrepresented measurable blunders.
- The output did not produce the accuracy sufficient for the diagnosis process

2.3 PROPOSED SYSTEM

Deep Learning is known to be learning a hierarchical set of representations such that it learns low mid and high-level features. Deep neural networks can adapt to more complex data sets. It's better in generalizing previously unseen data because of its multiple layers. Different algorithms use Deep Learning's fundamental expertise and use diverse datasets to train and test these algorithms. Like neurons in humans, deep learning has layers that help the model or algorithm learn and process the data. These layers process the data given to them as the input and learn by processing the input while traveling through the layers. When it passes out the last layer, an activation function is applied at last, and finally, we get the predicted output from the deep learning model. This gives us the training accuracy, and then when we take another similar type of dataset, we can predict or detect from the learned deep learning model whatsoever we want. Well, this is what deep learning does in simple working terms.

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

- Deep neural networks can adapt to more complex data sets. It's better in generalizing previously unseen data because of its multiple layers.
- we can predict or detect disease from the learned deep learning model.

2.4 FEASIBILITY STUDY

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

- Economic Feasibility
- Technical Feasibility
- Behavioural Feasibility

2.4.1 ECONOMIC FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIOURAL FEASIBILITY

Whatever we think need not be feasible. It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Operating system : windows, linux
- Processor : minimum intel i3
- Ram : minimum 4 gb
- Hard disk : minimum 250gb

2.5.2 SOFTWARE REQUIREMENTS

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system: Windows 7,8,10,11
- Python idel 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

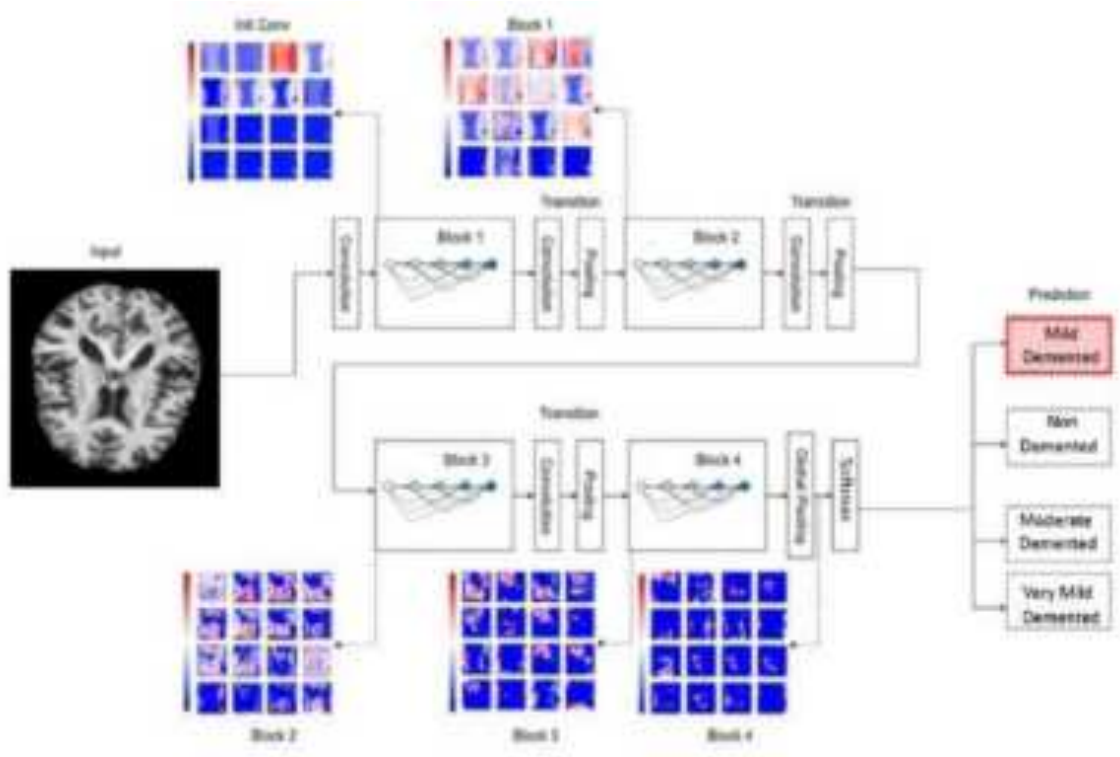


Figure 3.1: Project architecture for Prediction and Classification of Alzheimer Disease Severity Using Deep Learning Model

3.2 MODULES DESCRIPTION

The project is divided into 4 modules. The modular approach of the project is shown below in sequential manner

Module 1: Data Collection

The composition of the dataset.understand the relationship among different features. A plot of the core features and the entire dataset.The dataset is further split into 2/3 for training and 1/3 for testing the algorithms. Furthermore, in order to obtain a representative sample, each class in the full dataset is represented in about the right proportion in both the training and testing datasets. The various proportions of the training and testing datasets used in the paper. Collecting data allows you to capture a record of past events so that we can use data analysis to find recurring patterns. From those patterns, you build predictive models using machine learning algorithms that look for trends and predict future changes.Predictive models are only as good as the data from which they are built, so good data collection practices are crucial to developing high-performing models. The data need to be error-free (garbage in, garbage out) and contain relevant information for the task at hand. For example, a loan default model would not benefit from tiger population sizes but could benefit from gas prices over time. The easiest and fastest way to collect data for your ML model is to use an open-source dataset. Just like coding snippets, there are also thousands of open-source datasets available on the internet. They are free to use, easy to find and very time-effective to use. The only downside of using public datasets is that although they might contain a seemingly unlimited amount of rich, detailed data, you will probably need to clean according to your specific goal.

Module 2: Data Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency o the algorithm. The outliers have to be removed and also variable conversion need to be done. In order to overcoming these issues we use map function. To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as

the dataset.Dataset may be of different formats for different purposes, such as, if we want to create a machine learning model for business purpose, then dataset will be different with the dataset required for a liver patient. So each dataset is different from another dataset. To use the dataset in our code, we usually put it into a CSV file. However, sometimes, we may also need to use an HTML or xlsx file. In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset. In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model. Suppose, if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable

Module 3: Model Selection

Machine learning is about predicting and recognizing patterns and generate suitable results after understanding them. ML algorithms study patterns in data and learn from them. An ML model will learn and improve on each attempt. To gauge the effectiveness of a model, it’s vital to split the data into training and test sets first. So before training our models, we split the data into Training set which was 70% of the whole dataset and Test set which was the remaining 30%. Then it was important to implement a selection of performance metrics to the predictions made by our model.

Module 4: Predict the Results

The total number of features within the bank credit Defaulters dataset. However, not all have significant influence in determining the ability of a given customer in paying his/her loan or not. The designed system is tested with test set and the performance is assured. Evolution analysis refers to the description and model regularities or trends for objects whose behavior changes over time. Common metrics calculated from the confusion matrix are Precision; Accuracy. The mot important features since these features are to develop a predictive model using ordinary DenseNet model.

Algorithms:

DenseNet

DenseNet (Dense Convolutional Network) is an architecture that focuses on making the deep learning networks go even deeper, but at the same time making them more efficient to train, by using shorter connections between the layers. DenseNet is a convolutional neural network where each layer is connected to all other layers that are deeper in the network, that is, the first layer is connected to the 2nd, 3rd, 4th and so on, the second layer is connected to the 3rd, 4th, 5th and so on. This is done to enable maximum information flow between the layers of the network. To preserve the feed-forward nature, each layer obtains inputs from all the previous layers and passes on its own feature maps to all the layers which will come after it. Unlike Resnets it does not combine features through summation but combines the features by concatenating them. So the 'ith' layer has 'i' inputs and consists of feature maps of all its preceding convolutional blocks. Its own feature maps are passed on to all the next 'I-i' layers. This introduces '(I(I+1))/2' connections in the network, rather than just 'I' connections as in traditional deep learning architectures. It hence requires fewer parameters than traditional convolutional neural networks, as there is no need to learn unimportant feature maps. DenseNet consists of two important blocks other than the basic convolutional and pooling layers. they are the Dense Blocks and the Transition layers.

ResNet:

A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between.[1] An additional weight matrix may be used to learn the skip weights; these models are known as HighwayNets.[2] Models with several parallel skips are referred to as DenseNets.[3] In the context of residual neural networks, a non-residual network may be described as a plain network. A reconstruction of a pyramidal cell. Soma and dendrites are labeled in red, axon arbor in blue. (1) Soma, (2) Basal dendrite, (3) Apical dendrite, (4) Axon, (5) Collateral axon. There are two main reasons to add skip connections: to avoid the problem of vanishing gradients, or to mitigate the Degradation (accuracy saturation) problem; where adding more layers to a suitably deep model leads to higher training error.[1] During training, the weights adapt to mute the upstream layer[clarification needed], and amplify the previously-skipped layer. In the simplest case, only the weights for the adjacent layer's connection are adapted, with no explicit weights for the upstream layer. This works best when a single nonlinear layer is stepped over, or when the intermediate layers are all linear. If not, then an explicit weight matrix should be learned for the skipped connection (a HighwayNet should be used). Skipping effectively simplifies the network, using fewer layers in the initial training stages[clarification needed]. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Towards the end of training, when all layers are expanded, it stays closer to the manifold[clarification needed] and thus learns faster. A neural network without residual parts explores more of the feature space. This makes it more vulnerable to perturbations that cause it to leave the manifold, and necessitates extra training data to recover.

MobileNet

MobileNet uses depthwise separable convolutions. This convolution block was at first introduced by Xception. A depthwise separable convolution is made of two operations: a depthwise convolution and a pointwise convolution. A standard convolution works on the spatial dimension of the feature maps and on the input and output channels. It has a computational cost of $Df^2 * M * N * Dk^2$; with Df the dimension of the input feature maps, M and N the number of input and output channels, and Dk the kernel size. A depthwise convolution maps a single convolution on each input channel separately. Therefore its number of ouput channel is the same of the number of input channel. It computational cost is $Df^2 * M * Dk^2$.

3.3 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

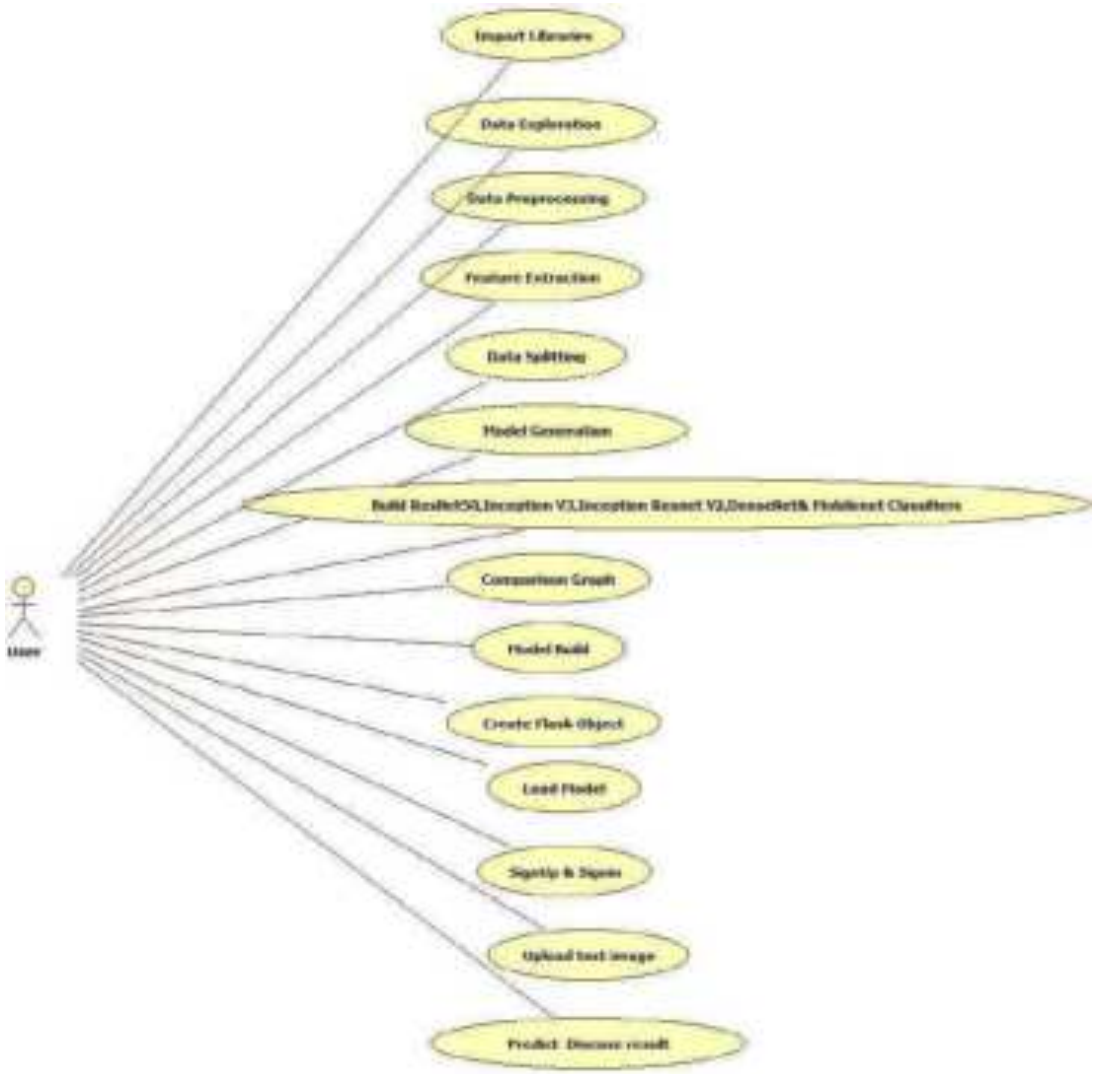


Figure3.2 Usecase diagram for Prediction and Classification of Alzheimer Disease Severity Using Deep Learning Model

3.4 CLASS DIAGRAM

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design.

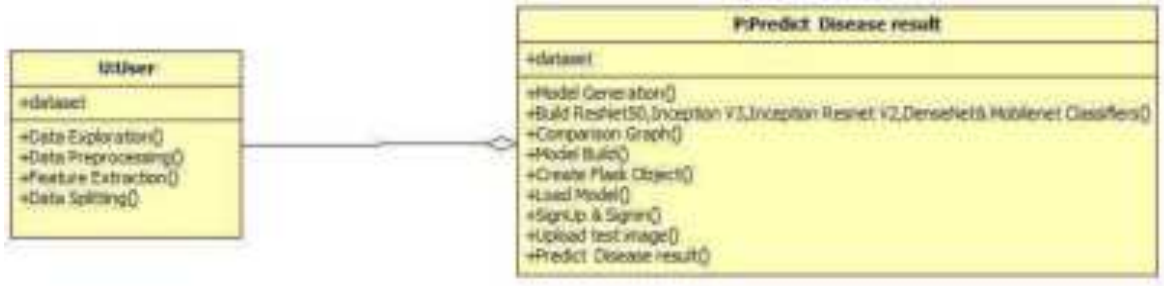


Figure3.3 Class diagram for Prediction and Classification of Alzheimer Disease Severity Using Deep Learning Model

3.5 SEQUENCE DIAGRAM

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows.

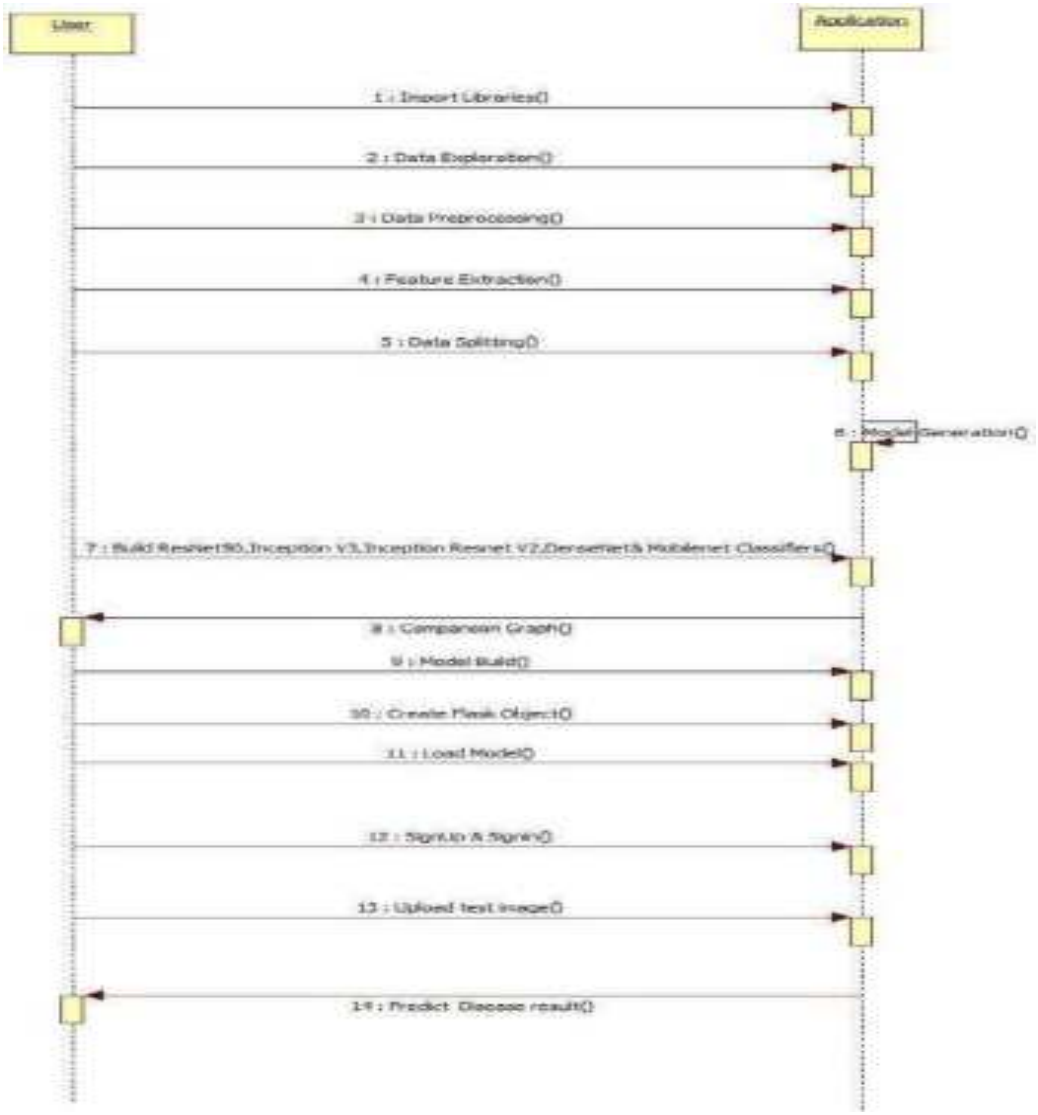


Figure3.4 Sequence diagram for Prediction and Classification of Alzheimer Disease Severity Using Deep Learning Model

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

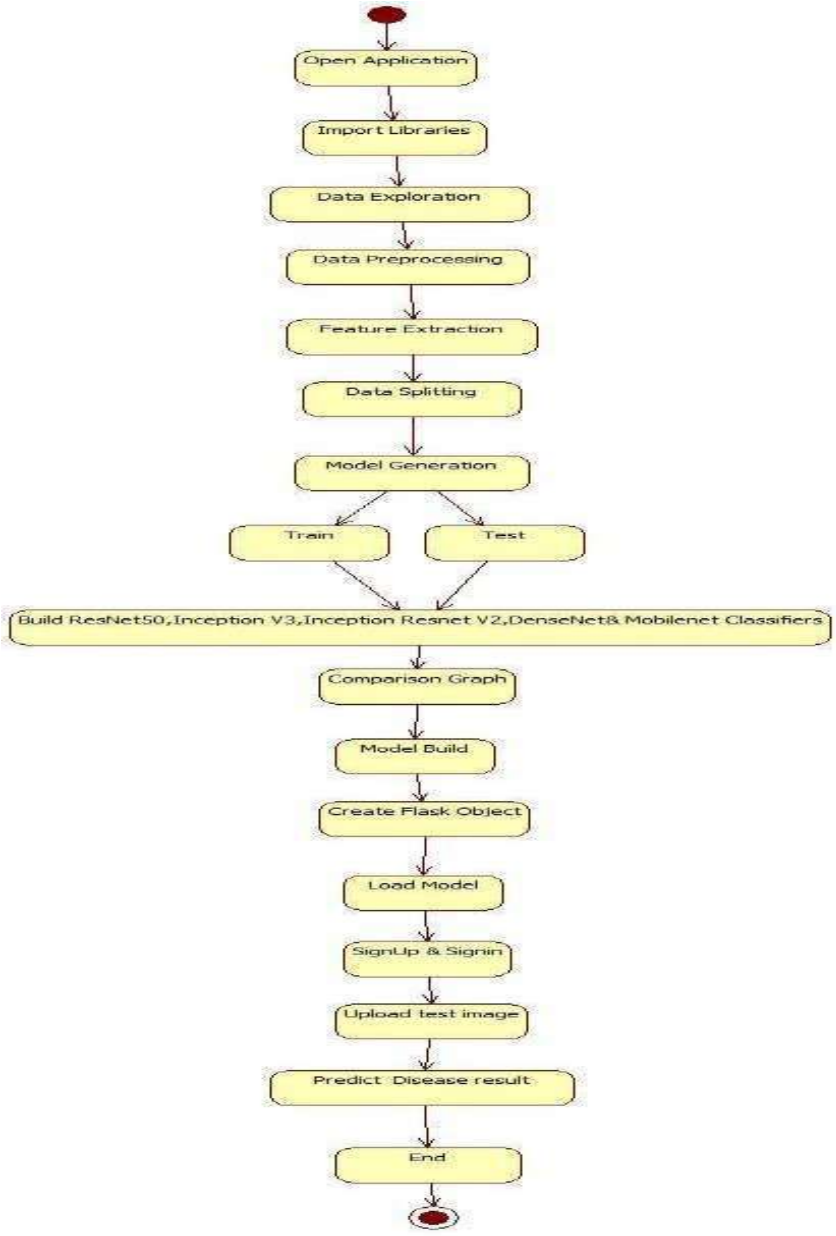


Figure3.5: Activity diagram for Prediction and Classification of Alzheimer Disease Severity Using Deep Learning Model

4. IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

```
import tensorflow
tensorflow.__version___
from tensorflow.keras.layers import Dense, Flatten, Input, Lambda
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import preprocess_input
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.models import Sequential
from glob import glob
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
#SEARCHING FOR P100
import os
import time
x=!nvidia-smi
count=0
```

```
for i in x:
    if "============" in i:
        count+=1
        break
    count+=1
if 'p100' in x[count].lower():
    print("found")
else:
    print(x[count])
    time.sleep(1)
    #os._exit(00)
import tensorflow
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
# Resizinig all the images to (224,224)
IMAGE_SIZE = [224,224]

train_path = 'Alzheimer_s Dataset/train'
test_path = 'Alzheimer_s Dataset/test'
# Scaling all the images between 0 to 1

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

# Performing only scaling on the test dataset

test_datagen = ImageDataGenerator(rescale=1./255)
train_set = train_datagen.flow_from_directory(train_path,
                                              target_size=(224,224),
```

```
                                        batch_size=32,
                                        class_mode = 'categorical')


test_set = test_datagen.flow_from_directory(test_path,
                                            target_size=(224,224),
                                            batch_size=32,
                                            class_mode='categorical')
# **ResNet50**
resnet = ResNet50(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
for layer in resnet.layers:
  layer.trainable = False
x = Flatten()(resnet.output)
prediction = Dense(4, activation='softmax')(x)
model = Model(inputs = resnet.inputs, outputs = prediction)
model.summary()
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
hist = model.fit(train_set, validation_data=test_set, epochs=200, steps_per_epoch=1, validation_steps=1,callbacks=[callback])
model.save('model1.h5')
import matplotlib.pyplot as plt


x=hist
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
```

```
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
# **For InceptionV3 with optimiser adam**
# create the base pre-trained model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
base_model = InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x2 = base_model.output
x2  = GlobalAveragePooling2D()(x2)
predictions = Dense(4, activation='softmax')(x2)

# this is the model we will train
models = Model(inputs=base_model.input, outputs=predictions)
models.summary()
for layer in base_model.layers:
    layer.trainable = False

models.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

```
hist = models.fit(train_set, validation_data=test_set, epochs=200, steps_per_epoch=len(train_set), validation_steps=len(test_set),callbacks=[callback])

models.save('model2.h5')

import matplotlib.pyplot as plt


x=hist

plt.figure(figsize=(20,10))

plt.subplot(1, 2, 1)

plt.suptitle('Optimizer : adam', fontsize=10)

plt.ylabel('Loss', fontsize=16)

plt.plot(x.history['loss'], label='Training Loss')

plt.plot(x.history['val_loss'], label='Validation Loss')

plt.legend(loc='upper right')


plt.subplot(1, 2, 2)

plt.ylabel('Accuracy', fontsize=16)

plt.plot(x.history['accuracy'], label='Training Accuracy')

plt.plot(x.history['val_accuracy'], label='Validation Accuracy')

plt.legend(loc='lower right')

plt.show()


# **INCEPTION RESNET V2**

from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2

from tensorflow.keras.applications.inception_resnet_v2 import preprocess_input

from tensorflow.keras.models import Model

inc=InceptionResNetV2(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)

x3 = Flatten()(inc.output)

predictionss = Dense(4, activation='softmax')(x3)

modelss = Model(inputs = inc.inputs, outputs = predictionss)
```

```
modelss.summary()

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

r2 = modelss.fit(train_set, validation_data=test_set, epochs=200, steps_per_epoch=len(train_set), validation_steps=len(test_set))

x=r2

modelss.save('model3.h5')

plt.figure(figsize=(20,10))

plt.subplot(1, 2, 1)

plt.suptitle('Optimizer : adam', fontsize=10)

plt.ylabel('Loss', fontsize=16)

plt.plot(x.history['loss'], label='Training Loss')

plt.plot(x.history['val_loss'], label='Validation Loss')

plt.legend(loc='upper right')


plt.subplot(1, 2, 2)

plt.ylabel('Accuracy', fontsize=16)

plt.plot(x.history['accuracy'], label='Training Accuracy')

plt.plot(x.history['val_accuracy'], label='Validation Accuracy')

plt.legend(loc='lower right')

plt.show()

# **MobileNet**

from tensorflow.keras.applications import MobileNet, MobileNetV2

mob = MobileNet(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)

x1= Flatten()(mob.output)

prediction1 = Dense(4, activation='softmax')(x1)

model12 = Model(inputs = mob.inputs, outputs = prediction1)

model12.summary()
```

```
model12.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

r1 = model12.fit(train_set, validation_data=test_set, epochs=200, steps_per_epoch=len(train_set), validation_steps=len(test_set))

model12.save('model4.h5')

import matplotlib.pyplot as plt


x=r1

plt.figure(figsize=(20,10))

plt.subplot(1, 2, 1)

plt.suptitle('Optimizer : adam', fontsize=10)

plt.ylabel('Loss', fontsize=16)

plt.plot(x.history['loss'], label='Training Loss')

plt.plot(x.history['val_loss'], label='Validation Loss')

plt.legend(loc='upper right')


plt.subplot(1, 2, 2)

plt.ylabel('Accuracy', fontsize=16)

plt.plot(x.history['accuracy'], label='Training Accuracy')

plt.plot(x.history['val_accuracy'], label='Validation Accuracy')

plt.legend(loc='lower right')

plt.show()

# **DenseNet121**

from tensorflow.keras.applications import DenseNet121, DenseNet169, DenseNet201, ResNet50V2,ResNet101V2,ResNet152V2

des121=DenseNet121(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)

x1= Flatten()(des121.output)

prediction1 = Dense(4, activation='softmax')(x1)

model1 = Model(inputs = des121.inputs, outputs = prediction1)
```

```
model1.summary()
model1.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
r1 = model1.fit(train_set, validation_data=test_set, epochs=200, steps_per_epoch=len(train_set), validation_steps=len(test_set))
model1.save('model_des.h5')
import matplotlib.pyplot as plt


x=r1
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')


plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
# **DenseNet169**
des169=DenseNet169(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
x1= Flatten()(des169.output)
prediction1 = Dense(4, activation='softmax')(x1)
model1 = Model(inputs = des169.inputs, outputs = prediction1)
model1.summary()
model1.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
r1 = model1.fit(train_set, validation_data=test_set, epochs=200, steps_per_epoch=len(train_set), validation_steps=len(test_set))

model1.save('model6.h5')

import matplotlib.pyplot as plt


x=r1

plt.figure(figsize=(20,10))

plt.subplot(1, 2, 1)

plt.suptitle('Optimizer : adam', fontsize=10)

plt.ylabel('Loss', fontsize=16)

plt.plot(x.history['loss'], label='Training Loss')

plt.plot(x.history['val_loss'], label='Validation Loss')

plt.legend(loc='upper right')


plt.subplot(1, 2, 2)

plt.ylabel('Accuracy', fontsize=16)

plt.plot(x.history['accuracy'], label='Training Accuracy')

plt.plot(x.history['val_accuracy'], label='Validation Accuracy')

plt.legend(loc='lower right')

plt.show()
```

5. SCREENSHOTS

5.1 SIGNUP OR SIGNIN TO ACCESS APPLICATION

Screenshot 5.1: Signup/Signin to access application

5.2 UPLOAD MRI IMAGE FOR CLASSIFICATION

Screenshot 5.2: Upload mri image for classification

5.3 RESULT AFTER CLASSIFICATION

Screenshot 5.3: Result after classification

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the

components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes**.**

6.3 TEST CASES

6.3.1 CLASSIFICATION

Test case id	Test case name	Purpose	Input	output
1	Classification test-1	To check the performance of the model	MRI image(very mild demented)	Very mild demented
2	Classification test-2	To check the performance of the model	MRI image(moderate demented)	Moderate demented
3	Classification test-3	To check the performance of the model	MRI image(Non demented)	Non demented
4	Classification test-4	To check the performance of the model	MRI image(mild demented)	Mild demented

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

Alzheimer's disease is the leading cause of dementia. This paper determines a prospective solution for detecting the disease at an early stage. The models used in this paper have successfully classified the images into the appropriate four classes and indeed provided us with promising results. We observe that DenseNet performs better than ResNet50,Inception V3,Inception Resnet V2 & Mobilenet. Further research is required to ensure that this particular model can be implemented in clinical settings, increasing the health care rate against this specific disease. Knowledge should be spread among people regarding this disease, and they should be encouraged to get themselves examined. We are currently working on deploying this model onto a website for better practical usage.

7.2 PROJECT FUTURE SCOPE

In the future, this model can also be tested on a larger dataset. We had only 52 and 12 images for training and testing in the current dataset for the 'Moderate Demented' class. The proposed model can help doctors diagnose Alzheimer's Disease more effectively and can be modified to identify other Neurodegenerative Diseases more automatically in the future.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 GITHUB REPOSITORY LINK

Link 1**:** https://github.com/sivasankar-3355/Major-Project

Link 2: https://github.com/AJMEERAMADHU/Major-project

Link 3: https://github.com/Aksh20/Major-Project

8.2 REFERENCES

[1] Suresha, Halebeedu Subbaraya, and Srirangapatna Sampathkumaran Parthasarathy. "Alzheimer Disease Detection Based on Deep Neural Network with Rectified Adam Optimization Technique using MRI Analysis." 2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC), pp. 1-6. IEEE, 2020.

[2] Deng, Lan, and Yuanjun Wang. "Hybrid diffusion tensor imaging feature-based AD classification." Journal of X-Ray Science and Technology Preprint, 2020, pp. 1-19.

[3] Khan, Afreen, and Swaleha Zubair. "An Improved Multi-Modal based Machine Learning Approach for the Prognosis of Alzheimer's disease." Journal of King Saud University-Computer and Information Sciences, 2020.

[4] Khan, Afreen, and Swaleha Zubair. "Usage Of Random Forest Ensemble Classifier Based Imputation And Its Potential In The Diagnosis Of Alzheimer's Disease." Int. J. Sci. Technol. Res. 8, no. 12, 2019, pp. 271- 275.

[5] Asim, Yousra, Basit Raza, Ahmad Kamran Malik, Saima Rathore, Lal Hussain, and Mohammad Aksam Iftikhar. "A multi-modal, multi-atlas- based approach for Alzheimer detection via machine learning." International Journal of Imaging Systems and Technology 28, no. 2, 2018, pp. 113-123.

8.3 WEBSITES

[1] www.medium.com

[2] www.edureka.co/blog/python-libraries/

[3] www.kaggle.com

9. JOURNAL

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:06/June-2022 Impact Factor- 6.752 www.irjmets.com

PREDICTION AND CLASSIFICATION OF ALZHEIMER DISEASE SEVERITY USING DEEP LEARNING MODEL

Mr. A Kiran Kumar[*1], A Sivasankar[*2], Akshitha Sabbineni[*3], Ajmeera Madhu[*4]

[*1]Assistant Professor, Computer Science And Engineering, CMR Technical Campus, Hyderabad, Telangana, India.

[*2,3,4]Student, Computer Science And Engineering, CMR Technical Campus, Hyderabad, Telangana, India.

ABSTRACT

Alzheimer's disease is an unrepairable degenerative brain disease. Every four seconds, someone in the world is diagnosed with Alzheimer's disease. The outcome is irreversible, and it leads to a life-threatening condition. As a result, it's crucial to catch the disease early on. The leading cause of dementia is Alzheimer's disease. Dementia causes a reduction in reasoning abilities and interpersonal coping skills, which affects people's ability to function independently. The patient will forget recent events in the early stages. If the illness progresses, they will gradually forget whole events. It's critical to get a diagnosis of the disease as soon as possible. This research offers a model that uses brain MRI sample pictures as input and output to assess whether a person has mild, moderate, or no Alzheimer's disease. We are using the ResNet50,Inception V3,Inception ResNet V2,DenseNet and MobileNet architectures for this classification, providing a comparative analysis of which architecture shows promising results.

Keywords: Alzheimer Disease, Dementia, MRI Images, Resnet, Densenet, Mobilenet, Mild Demented , Moderate Demented.

I. INTRODUCTION

The system architecture shows us a conceptual and behavioral view of the system. It is just a view that shows us how the database is used in taking the dataset and then how this data is used up in our project modules to train the different models. In the architecture diagram above, The data is taken from the training dataset and then delivered to the models, as can be seen. Then it is validated against the test dataset to get the testing or validation accuracy. After the accuracy is compared, the diseased images are taken from the dataset, the classification done in four classes namely Mild Demented, Moderate Demented, Non Demented and Very Mild Demented. Also, the architecture diagram shows us the various modules working together in the project and how they are integrated to provide us the desired output, and how all the modules need to be interconnected to make the project work in unison.

II. DEEP LEARNING PROCESS

This research compares two state-of-the-art deep learning models' detection accuracy in detecting Alzheimer's disease in an MRI image. The Keras module of tensor flow, an opensource library for implementing deep learning models, is used to implement VGG19. The data was supplemented and fed into the model using the Image Data Generator tool. The training consisted of a batch size of 128, with 50 epochs with early stopping. Similarly, the Keras module is used to implement the Densenet Model, and the data is loaded into the densenet model via the Image Data Generator function. The densenet model is trained using a batch of 128 images each. Both models are tested on a total of 2067 MRI images after being trained on 3048 MRI images divided into four groups. Google colab was used for the entire project.

STEP 1: Data Collection

The composition of the dataset.understand the relationship among different features. A plot of the core features and the entire dataset.The dataset is further split into 2/3 for training and 1/3 for testing the algorithms. Furthermore, in order to obtain a representative sample, each class in the full dataset is represented in about the right proportion in both the training and testing datasets. The various proportions of the training and testing datasets used in the paper.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:06/June-2022 Impact Factor- 6.752 www.irjmets.com

STEP 2: Data Preprocessing

There is a chance that the data obtained contains missing values, which could lead to inconsistencies. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. Outliers must be eliminated, and variables must be converted. We utilise the map function to solve these problems.

STEP 3: Data Visualization

Graphs are plotted upon the preprocessed data to have proper visualization of the data collected and preprocessed. Data visualization is done to know the balance of the data among our various classes in our dataset. Additionally, data visualisation can be used to partition data.

STEP 4: Data Splitting

The preprocessed data is evenly splitted for testing and training purposes. 70% of the data in the dataset is used to train the model, while 30% of the data is utilised to test the model.

STEP 5: Model Selection

Machine learning is about predicting and recognizing patterns and generate suitable results after understanding them. Algorithms that use machine learning look for patterns in data and learn from them. An ML model will learn and improve on each attempt. To determine a model's effectiveness, the data must first be divided into training and test sets. So before training our models, we split the data into Training set which was 70% of the whole dataset and Test set which was the remaining 30%. It was therefore necessary to apply a variety of performance indicators to our model's predictions.

STEP 6: Predict The Results

The amount of characteristics in the Alzheimer's dataset in total. However, not all have significant influence in determining the ability of a given customer in paying his/her loan or not. The designed system is tested with test set and the performance is assured. The term "evolution analysis" refers to the description and modelling of regularities or patterns in the behaviour of objects that evolve through time. Common metrics calculated from the confusion matrix are Precision; Accuracy. The mot important features since these features are to develop a predictive model using ordinary DenseNet model.

III. PROJECT ARCHITECTURE

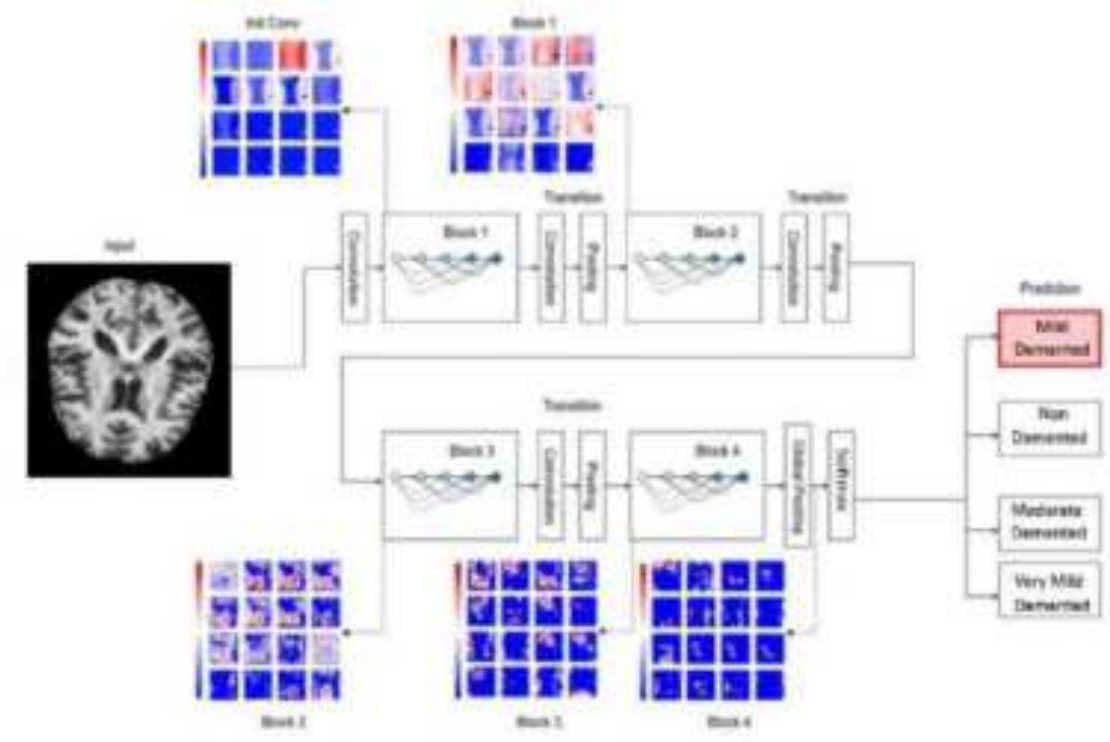


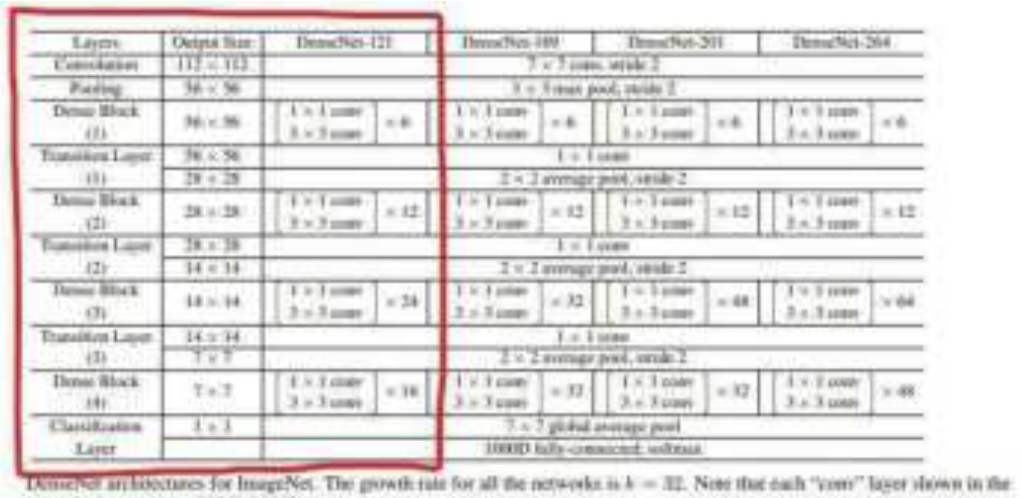
Figure 1: Deep Learning-Based Alzheimer's Severity Classification Architecture Diagram.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:06/June-2022 Impact Factor- 6.752 www.irjmets.com

IV. DENSENET, RESNET, MOBILENET USED IN DEEP LEARNING

Densenet:

DenseNet (Dense Convolutional Network) is an architecture that focuses on making the deep learning networks go even deeper, but at the same time making them more efficient to train, by using shorter connections between the layers. DenseNet is a convolutional neural network with each layer connected to the ones below it, that is, the first layer is connected to the 2nd, 3rd, 4th and so on, the second layer is connected to the 3rd, 4th, 5th and so on. This is done to ensure that the maximum amount of data can move between the network's tiers. Each layer receives inputs from all previous layers to maintain the feed-forward character and passes on its own feature maps to all the layers which will come after it. Unlike Resnets it does not combine features through summation but combines the features by concatenating them. So the 'ith' layer has 'i' inputs and consists of feature maps of all its preceding convolutional blocks. Its own feature maps are passed on to all the next 'I-i' layers. This introduces '(I(I+1))/2' connections in the network, rather than just 'I' connections as in traditional deep learning architectures. It hence requires fewer parameters than traditional convolutional neural networks, as there is no need to learn unimportant feature maps. Aside from the fundamental convolutional and pooling layers, DenseNet has two crucial pieces they are the Dense Blocks and the Transition layers.



DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

Resnet:

A residual neural network (ResNet) is a type of artificial neural network (ANN) that is based on components discovered in cerebral cortex pyramidal cells. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between.[1] An additional weight matrix may be used to learn the skip weights; these models are known as HighwayNets.[2] Models with several parallel skips are referred to as DenseNets.[3] In the context of residual neural networks, a non-residual network may be described as a plain network. A reconstruction of a pyramidal cell. Soma and dendrites are labeled in red, axon arbor in blue. (1) Soma, (2) Basal dendrite, (3) Apical dendrite, (4) Axon, (5) Collateral axon. There are two main reasons to add skip connections: to avoid the problem of vanishing gradients, or to mitigate the Degradation (accuracy saturation) problem; where adding more layers to a suitably deep model leads to higher training error.[1] During training, the weights adapt to mute the upstream layer[clarification needed], and amplify the previously-skipped layer. In the simplest case, only the weights for the adjacent layer's connection are adapted, with no explicit weights for the upstream layer. This works best when a single nonlinear layer is stepped over, or when the intermediate layers are all linear. Skipping effectively simplifies the network, using fewer layers in the initial training stages[clarification needed]. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Towards the end of training, when all layers are expanded, it stays closer to the manifold[clarification needed] and thus learns faster. A neural network with no residual

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:06/June-2022 Impact Factor- 6.752 www.irjmets.com

components explores more of the feature space than one with residual parts. This makes it more vulnerable to perturbations that cause it to leave the manifold, and necessitates extra training data to recover.

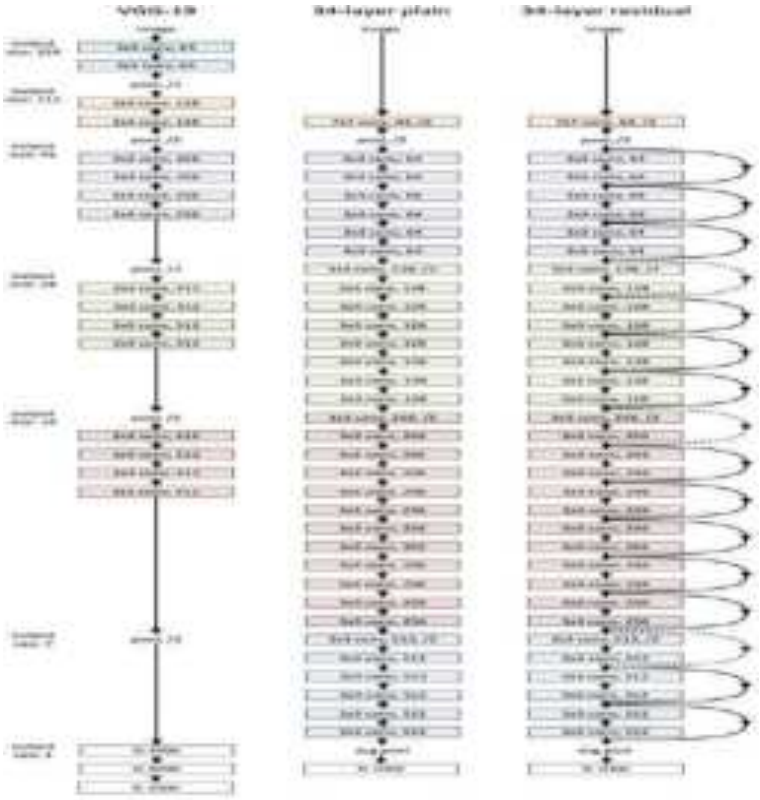
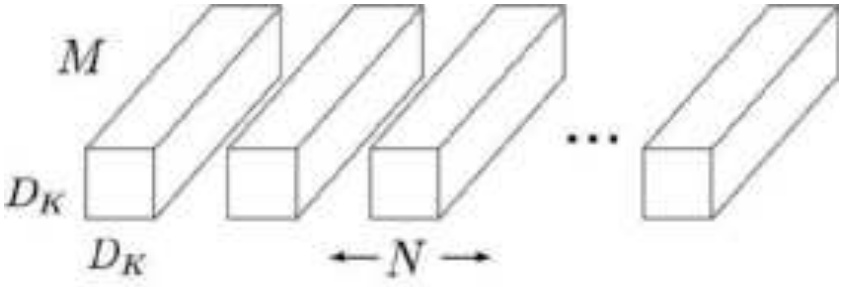


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Mobilenet:

MobileNet uses depthwise separable convolutions. This convolution block was at first introduced by Xception. A depthwise separable convolution is made up of two procedures: a depthwise convolution and a pointwise convolution. The spatial dimension of the feature maps, as well as the input and output channels, are all used in a conventional convolution. It has a computational cost of $Df^2 * M * N * Dk^2$; with Df the dimension of the input feature maps, M and N the number of input and output channels, and Dk the kernel size. A depthwise convolution maps a single convolution on each input channel separately. Therefore its number of ouput channel is the same of the number of input channel. It computational cost is $Df^2 * M * Dk^2$.



(a) Standard Convolution Filters

V. CONCLUSION

Alzheimer's disease is the most frequent cause of dementia. This paper identifies a potential technique for early detection of the condition. The models utilised in this paper effectively categorised the photos into the four

groups we needed, and the findings were promising. DenseNet outperforms ResNet50, Inception V3, Inception Resnet V2, and Mobilenet in our tests. More study is needed to ensure that this approach can be used in clinical settings, hence boosting the rate of health care for this disease. People should be educated about this disease, and they should be urged to get themselves checked out. We're now working on putting this model on a website so that it may be more easily used. This model can be evaluated on a larger dataset in the future. We only had 52 and 12 pictures for training and testing in the present dataset for the 'Moderate Demented' class, respectively. The suggested approach can assist doctors in more effectively diagnosing Alzheimer's Disease, and it can be upgraded in the future to automatically detect other neurodegenerative diseases.

VI. REFERENCES

[1] Suresha, Halebeedu Subbaraya, and Srirangapatna Sampathkumaran Parthasarathy. "Alzheimer Disease Detection Based on Deep Neural Network with Rectified Adam Optimization Technique using MRI Analysis." 2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC), pp. 1-6. IEEE, 2020.

[2] Deng, Lan, and Yuanjun Wang. "Hybrid diffusion tensor imaging feature-based AD classification." Journal of X-Ray Science and Technology Preprint, 2020, pp. 1-19.

[3] Khan, Afreen, and Swaleha Zubair. "An Improved Multi-Modal based Machine Learning Approach for the Prognosis of Alzheimer's disease." Journal of King Saud University-Computer and Information Sciences, 2020.

[4] Khan, Afreen, and Swaleha Zubair. "Usage Of Random Forest Ensemble Classifier Based Imputation And Its Potential In The Diagnosis Of Alzheimer's Disease." Int. J. Sci. Technol. Res. 8, no. 12, 2019, pp. 271-275.

[5] Asim, Yousra, Basit Raza, Ahmad Kamran Malik, Saima Rathore, Lal Hussain, and Mohammad Aksam Iftikhar. "A multi-modal, multi-atlas- based approach for Alzheimer detection via machine learning." International Journal of Imaging Systems and Technology 28, no. 2, 2018, pp. 113-123.

[6] Alam, Saruar, Goo-Rak Kwon, and Alzheimer's Disease Neuroimaging Initiative. "Alzheimer disease classification using KPCA, LDA, and multi-kernel learning SVM." International Journal of Imaging Systems and Technology 27, no. 2, 2017, pp. 133-143.

[7] Brookmeyer, R., Grey, S. & Kawash, C. Projections of Alzheimer's disease in the United States and the public health impact of delaying disease onset. Am. J. Public Health 88, 1337–1342 (1998).

[8] Lama, Ramesh Kumar, Jeonghwan Gwak, Jeong-Seon Park, and SangWoong Lee. "Diagnosis of Alzheimer's disease based on structural MRI images using a regularized extreme learning machine and PCA features." Journal of healthcare engineering 2017, 2017.

[9] Bryan, R. Nick. "Machine learning applied to Alzheimer disease." , 2016, pp. 665-668.

[10] Escudero, Javier, Emmanuel Ifeachor, John P. Zajicek, Colin Green, James Shearer, Stephen Pearson, and Alzheimer's Disease Neuroimaging Initiative. "Machine learning-based method for personalized and costeffective detection of Alzheimer's disease." IEEE transactions on biomedical engineering 60, no. 1, 2012, pp. 164-168

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 06/40600115360

Date: 19/06/2022

Certificate of Publication

This is to certify that author "**A Sivasankar**" with paper ID "**IRJMETS40600115360**" has published a paper entitled "PREDICTION AND CLASSIFICATION OF ALZHEIMER DISEASE SEVERITY USING DEEP LEARNING MODEL" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 06, June 2022**

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 06/40600115360

Date: 19/06/2022

Certificate of Publication

This is to certify that author "**Akshitha Sabbineni**" with paper ID "**IRJMETS40600115360**" has published a paper entitled "PREDICTION AND CLASSIFICATION OF ALZHEIMER DISEASE SEVERITY USING DEEP LEARNING MODEL" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 06, June 2022**

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 4/Issue 06/40600115360

Date: 19/06/2022

Certificate of Publication

This is to certify that author "**Ajmeera Madhu**" with paper ID "**IRJMETS40600115360**" has published a paper entitled "PREDICTION AND CLASSIFICATION OF ALZHEIMER DISEASE SEVERITY USING DEEP LEARNING MODEL" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 4, Issue 06, June 2022**

Editor in Chief

IRJMETS Impact Factor 6.752

We Wish For Your Better Future

www.irjmets.com