

Final Project Report

SmartLender - Cognitive Care: Early Intervention for Alzheimer's Disease

1.Introduction

1.1Project overviews

Alzheimer's disease (AD) is a progressive and irreversible neurological disorder that affects the brain, leading to memory loss, cognitive impairment, and changes in behavior and personality. It is the most common cause of dementia among older adults and is characterized by the buildup of abnormal protein deposits in the brain, including amyloid plaques and tau tangles.

1.2Objectives

Is to develop an AI model for early and accurate detection of Alzheimer's disease. Improve the sensitivity of cognitive assessments by leveraging transfer learning techniques. Facilitate early intervention for individuals at risk for Alzheimer's disease.

2.Project Initialization and Planning Phase

2.1.Define Problem Statement

Alzheimer's Disease is a neurological brain disorder that damages the cells in brain and reduce the ability of the brain from the regular activities. It is a representation of the most common form of adult-onset dementias. Earlier detection of Alzheimer's disease can be more helpful in predetermining the symptomatic conditions of patients suffering with this problem. By diagnosing the consequences of this disease, with the help of medical scan images, it would be more useful in classifying the patients whether they are suffering from this deadly disease. Machine Learning tends to be more beneficial in diagnosing diseases and implementation of this technique, to Magnetic Resonance Imaging (MRI) inputs in identification of Alzheimer's disease, resulted in faster prediction of the disease and in the contribution of the evolution of the disease. Carrying out this technique, it is possible to diagnose and predict the individual dementia of adults by screening data of Alzheimer's disease and inducing Machine Learning classifiers. This work focuses on building an evolving framework to detect Alzheimer's disease efficiently with the help of neuroimaging technologies and prediction at a very earlier stage by using the data stacked up for Alzheimer's disease patients.

REGERENCE LINK:- [Click Here](#)

2.2: Project Proposal (Proposed Solution)

This project proposes to develop a machine learning system that can automatically detect Alzheimer's disease from brain MRI scans. The system will be trained on a large dataset of labeled MRI images from patients with and without Alzheimer's. Advanced deep learning techniques like convolutional neural networks will be employed to analyze the 3D brain scan data and identify patterns and biomarkers indicative of Alzheimer's pathology. Specific emphasis will be placed on detecting early signs of the disease before significant cognitive impairment occurs. The trained model will take a new patient's MRI scan as input and output a prediction of whether Alzheimer's is present along with a probability score. This could provide a powerful diagnostic tool to aid clinicians in making more accurate and timely Alzheimer's assessments. Robust validation will be performed on held-out test data to evaluate the system's performance. This approach aims to achieve early and accurate detection of Alzheimer's, potentially surpassing the limitations of current methods. The benefits could be significant, leading to improved diagnostic efficiency, reduced healthcare costs associated with delayed diagnosis, and ultimately paving the way for personalized treatment plans based on the early identification of Alzheimer's disease.

REFERENCE LINK:- [Click Here](#)

2.3. Initial Project Planning

The project planning template provided outlines a project titled Cognitive Care: Early Intervention for Alzheimer's Disease. The project involves various sprints with defined tasks, team members, priorities, and timelines. It includes phases like Data Collection, Image Preprocessing, Model Building & Testing, Application building. Each sprint has specific user stories, points, and team members assigned to different tasks. The project progresses through phases such as defining problem statements, project proposals, data collection, model training, optimization, and ends with creating executable files, documentation, and demonstration. The template emphasizes a structured approach to project management, ensuring clear objectives, team collaboration, and a timeline for completion.

REFERENCE LINK:- [Click Here](#)

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

The dataset for "Cognitive Care: Early Intervention For Alzheimer's Disease" is sourced from Kaggle. It provides a comprehensive dataset of cognitive, behavioral, and biomarker data from individuals at risk of developing Alzheimer's disease or in the early stages of the disease, with the goal of developing effective early intervention strategies.

REFERENCE LINK:- [Click Here](#)

3.2.Data Quality Report

The collected datasets provide comprehensive and accurate cognitive, behavioral, and biomarker data from diverse participants at various stages of cognitive decline. Standard protocols ensure consistency in data collection across sites. Robust data management maintains integrity and protects privacy. Metadata is detailed to enable proper analysis. The datasets include both recent and longitudinal data for timely analysis of disease progression. Established data sources facilitate accessibility and reproducibility. Continuous quality improvement processes are in place to address any issues and incorporate feedback. Overall, the data quality meets high standards for supporting early intervention research in Alzheimer's disease.

REFERENCE LINK:-[Click Here](#)

3.3.Data Preprocessing

The data exploration and preprocessing stages are crucial for ensuring data quality, identifying potential issues, and preparing the datasets for effective modeling and analysis in the context of early intervention strategies for Alzheimer's disease.

Preprocessing includes handling missing values, scaling, and encoding categorical variables. These crucial steps enhance data quality, ensuring the reliability and effectiveness

REFERENCE LINK:-[Click Here](#)

4.Model Development Phase

The Model Development Phase entails crafting a predictive model for Early Intervention For Alzheimer's Disease. It encompasses strategic feature selection, evaluating and selecting models (Xception, DenseNet, MobileNet), initiating training with code, and rigorously validating and assessing model performance for informed decision-making in the lending process. The Model Development Phase is critical for translating the collected data into actionable insights and decision support tools for early intervention and personalized care in Alzheimer's disease management.

4.1.Model Selection Report

For this project on detecting Alzheimer's disease from 3D brain MRI scan data, we propose using the DenseNet169 convolutional neural network architecture, a powerful deep learning model well-suited for image analysis tasks like this. Some key advantages of DenseNet169 are its design for 3D data through adaptable 3D convolutional layers highly effective at capturing complex 3D spatial patterns, proven state-of-the-art performance on many visual recognition benchmarks demonstrating high accuracy with the 169-layer variant capable of modeling intricate brain structures, parameter efficiency through feature reuse allowing training very deep networks effectively on moderately-sized 3D datasets, and interpretability providing insights into what the model detects as Alzheimer's indicators, though greater memory consumption can be mitigated through techniques like gradient checkpointing. Overall, DenseNet169

balances high predictive capabilities with architectural properties well-suited to analyzing 3D MRI data, making it an excellent accurate and transparent model choice compared to other 3D CNNs.

REFERENCE LINK:-[Click Here](#)

4.2.Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code implemented a range of machine learning algorithms, including logistic regression, random forests, and deep neural networks, using the selected features from cognitive assessments, neuroimaging data, genetic markers, and demographic/clinical factors. Appropriate data preprocessing, feature scaling, and cross-validation techniques (e.g., stratified k-fold) were employed. The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots

REFERENCE LINK:-[Click Here](#)

5.Model Optimization and Tuning Phase

5.1.Tuning Documentation

The Hyperparameter Tuning Documentation for the "Cognitive Care: Early Intervention For Alzheimer's Disease" project involves tuning the DenseNet169 model with specific hyperparameters to optimize its performance. The tuned hyperparameters include the input size of the image, the optimizer function with Adam and learning rate, the number of classes to be classified, and the variable to save the model. Additionally, parameters for the callback function, the number of epochs to run, the dataset to use, and other settings are adjusted to enhance the model's efficiency and accuracy during training and evaluation¹. This meticulous tuning process aims to fine-tune the model for peak performance, ensuring that it can effectively classify ship images with high accuracy and efficiency.

REFERENCE LINK:-[Click Here](#)

5.2.Final Model Selection Justification

DenseNet169 can be a good candidate model for initial exploration due to its strong feature extraction capabilities. DenseNet169 has a complex architecture with many layers, potentially leading to overfitting, especially with limited medical datasets. DenseNet's architecture promotes feature reuse and alleviates the vanishing gradient problem, potentially leading to robust feature extraction from MRI images. Pre-trained DenseNet169 on large image datasets can be fine-tuned for AD prediction, leveraging its ability to learn generalizable image features. DenseNet based multi-class categorization network optimized with a focused loss to assess the clinical stage of the predicted brain. The final DenseNet169 algorithm class categorization of AD diagnosis includes mild

demented, moderate demented, non-demented, and very mild demented. As a result, the unified framework allows for the simultaneous optimization of classifierModeling and pattern recognition (from communal space to labeled space and from original feature space to common space,respectively)

REGERENCE LINK:- [Click Here](#)

6.Results

6.1.Output Screenshots

```

# Test Case 1: Non-Demented
dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}

img = load_img('../content/Alzheimer_s Dataset/test/NonDemented/26 (69).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)

# Load the model
model = load_model('../content/model.h5')

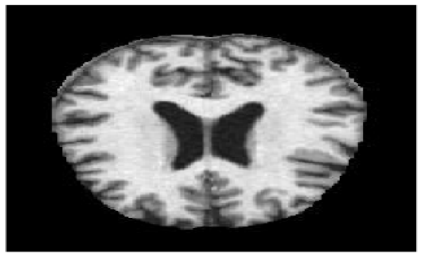
# Predict the class
answer = model.predict(img)

# Extract the class label and name
class_label = np.argmax(answer)
class_name = idc[class_label]

# Calculate and print the probability
probability = round(np.max(answer)*100,2)
print(probability, '% chances are there that the image is', class_name)

1/1 [=====] - 4s 4s/step
45.12 % chances are there that the image is NonDemented

```



```

# Test Case 2: MildDemented

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

img = load_img('/content/Alzheimer_s Dataset/test/MildDemented/26 (23).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)

# Load the model
model = load_model('/content/model.h5')

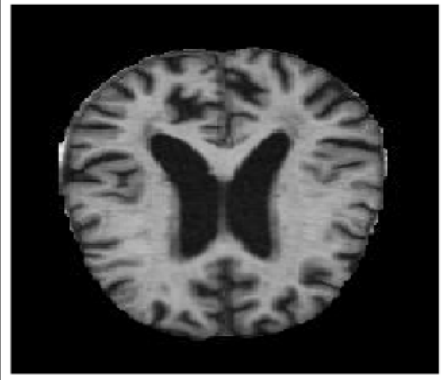
# Predict the class
answer = model.predict(img)

# Extract the class label and name
class_label = np.argmax(answer)
class_name = idc[class_label]

# Calculate and print the probability
probability = round(np.max(answer)*100,2)
print(probability, '% chances are there that the image is', class_name)

```

1/1 [=====] - 55 55/step
69.94 % chances are there that the image is MildDemented



```

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

img = load_img('/content/Alzheimer_s Dataset/test/NorDemented/26 (49).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)

# Load the model
model = load_model('/content/model.h5')

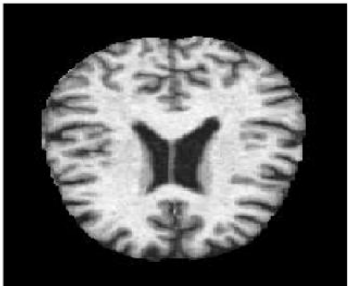
# Predict the class
answer = model.predict(img)

# Extract the class label and name
class_label = np.argmax(answer)
class_name = idc[class_label]

# calculate and print the probability
probability = round(np.max(answer)*100,2)
print(probability, '% chances are there that the image is', class_name)

```

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7ebcb29b37f0> triggered tf.funct
1/1 [=====] - 45 45/step
86.88 % chances are there that the image is NorDemented





[Home](#) [About](#) [Services](#) [Team](#) [Contact](#)

BrainVitals: Detecting Alzheimer's, Empowering Lives

Concerned about memory loss? You're not alone. At BrainVitals, we understand the importance of early detection and support for cognitive health. We offer a welcoming and comfortable environment where you can learn more about memory screenings and explore options for maintaining brain function.



[Review Your Reports](#)

[Learn More](#)



[Home](#) [About](#) [Services](#) [Team](#) [Contact](#)



Easy to Test MRI Reports

Get Your Easy-to-Read MRI Report Today! This empowers you to be more informed.



MIND diet for brain health

May help improve memory and cognitive function. Learn more about the MIND diet.



Sharpen your mind with brain training.

Our brain training exercises are designed to be engaging and effective.



About Our Services

Alzheimer's Detection Using MRI Reports, Diet Plan for Brain Health, Brain Exercises for a Healthy Mind, We are a team of passionate individuals dedicated to empowering early intervention in Alzheimer's Disease. Our mission is to leverage cutting-edge technology to identify potential risks for AD and provide users with the resources they need to take a proactive approach to their brain health.

500+

Happy Customers

10+

Qualified Dietitians

98%

Exact MRI Report Prediction

55+

Effective Brain Exercises



Recommend Diet Plans



Mediterranean Diet

Fruits and vegetables (especially leafy greens), Whole grains, Legumes (beans and lentils) Fish and seafood (rich in omega-3 fatty acids), Healthy fats like olive oil. limits: Red meat Processed foods Sugary drinks Added sugar



MIND Diet

This combines elements of the Mediterranean and DASH diets, focusing on: Leafy green vegetables (e.g., spinach, kale), Berries (known for antioxidants), Nuts (almonds, walnuts), Whole grains (brown rice, quinoa), Fish (at least once a week). limits: Red meat Sweets and pastries Cheese Fried foods Processed foods



DASH Diet (Dietary Approaches to Stop Hypertension)

Alzheimer's, this diet can be helpful because it promotes good blood flow, which is crucial for brain health. It emphasizes: Fruits and vegetables, Low-fat dairy products, Whole grains, Lean protein sources (fish, poultry), Nuts and seeds in moderation. It limits: Saturated and trans fats Red meat Added sugar and sodium



The Team Members



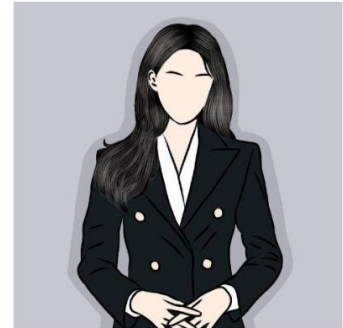
Sakshi Awachare

Member no.1



Vaishnavi Parab

Member no.2



Sakshi Ubale

Member no.3



**Request your
appointment and start
your !**

[Request Appointment](#)



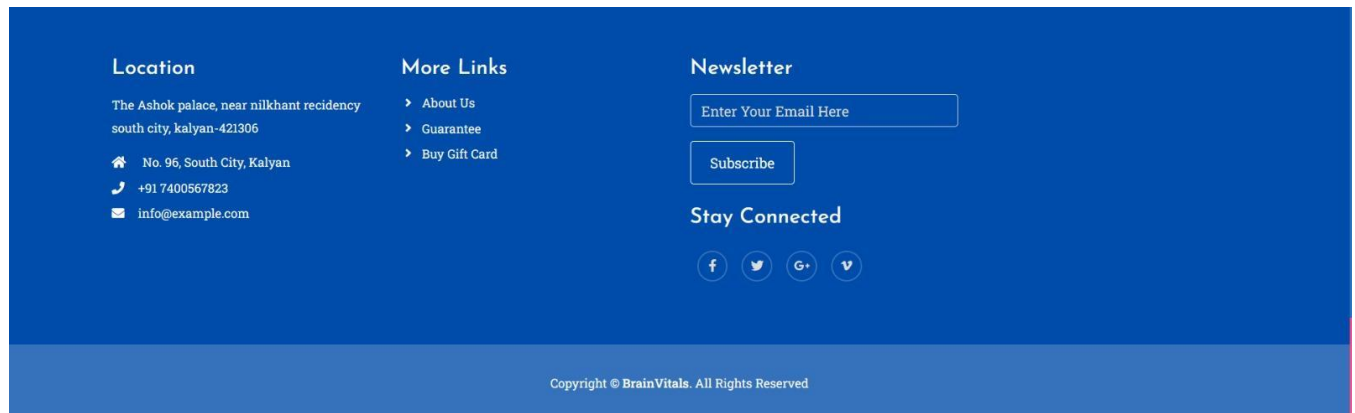
Location

The Ashok palace, near nilkhant recidency
south city, kalyan-421306

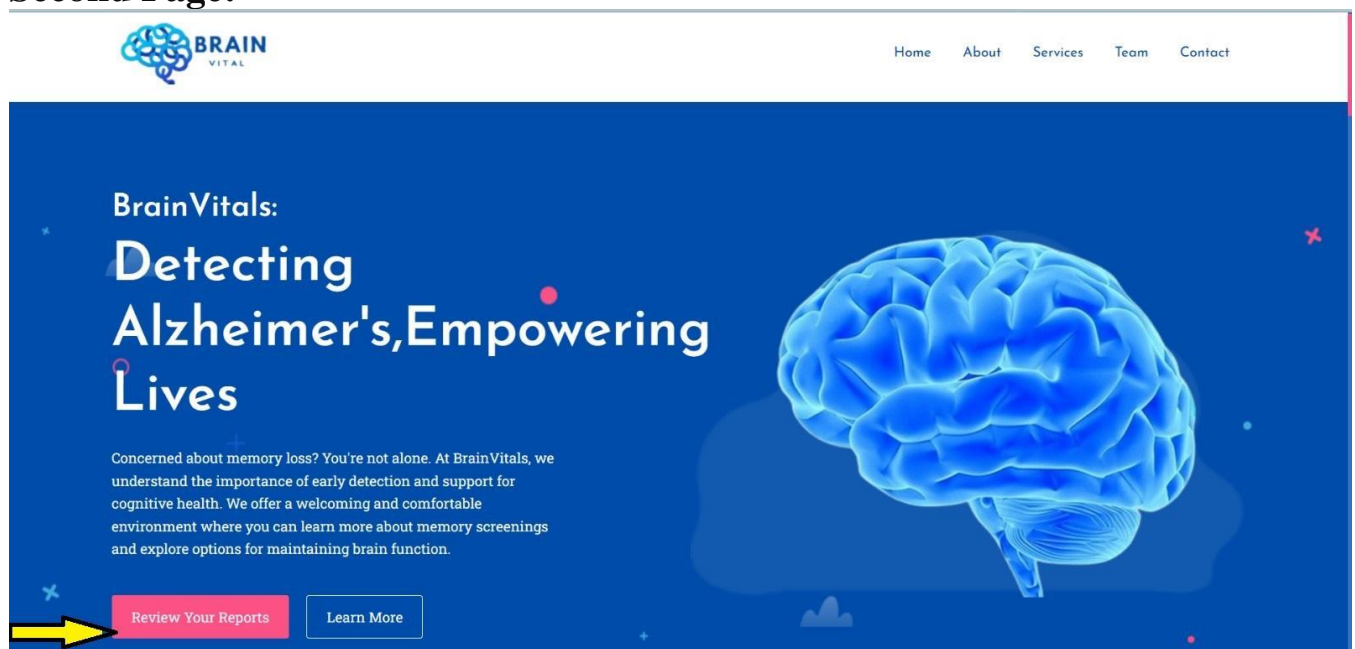
More Links

- [About Us](#)
- [Guarantee](#)

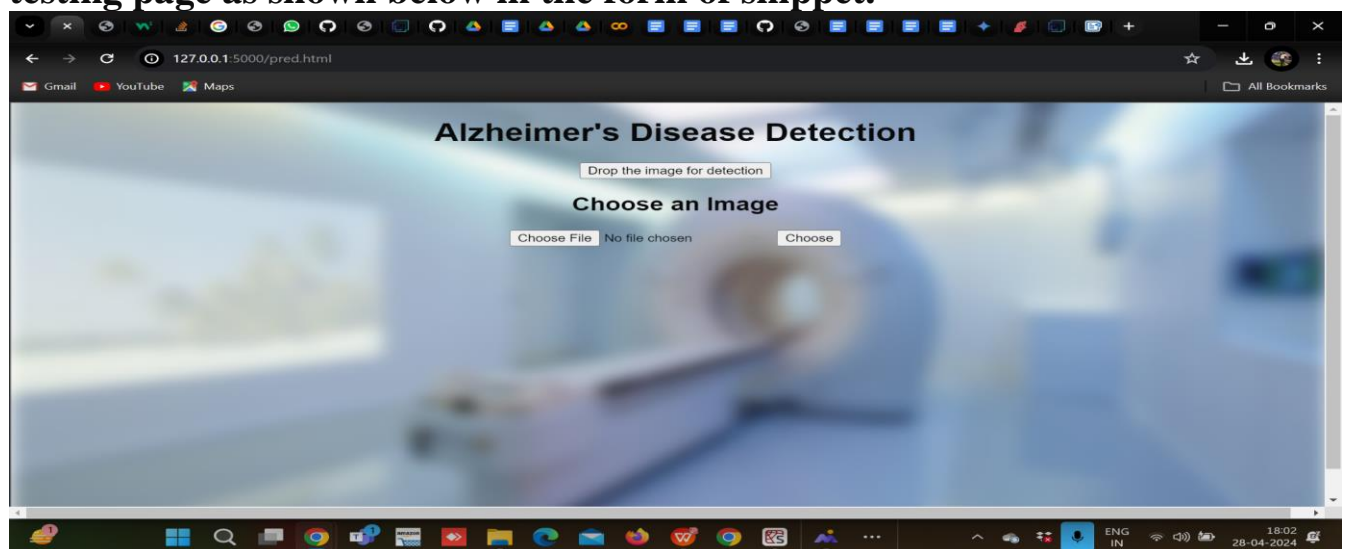
Newsletter

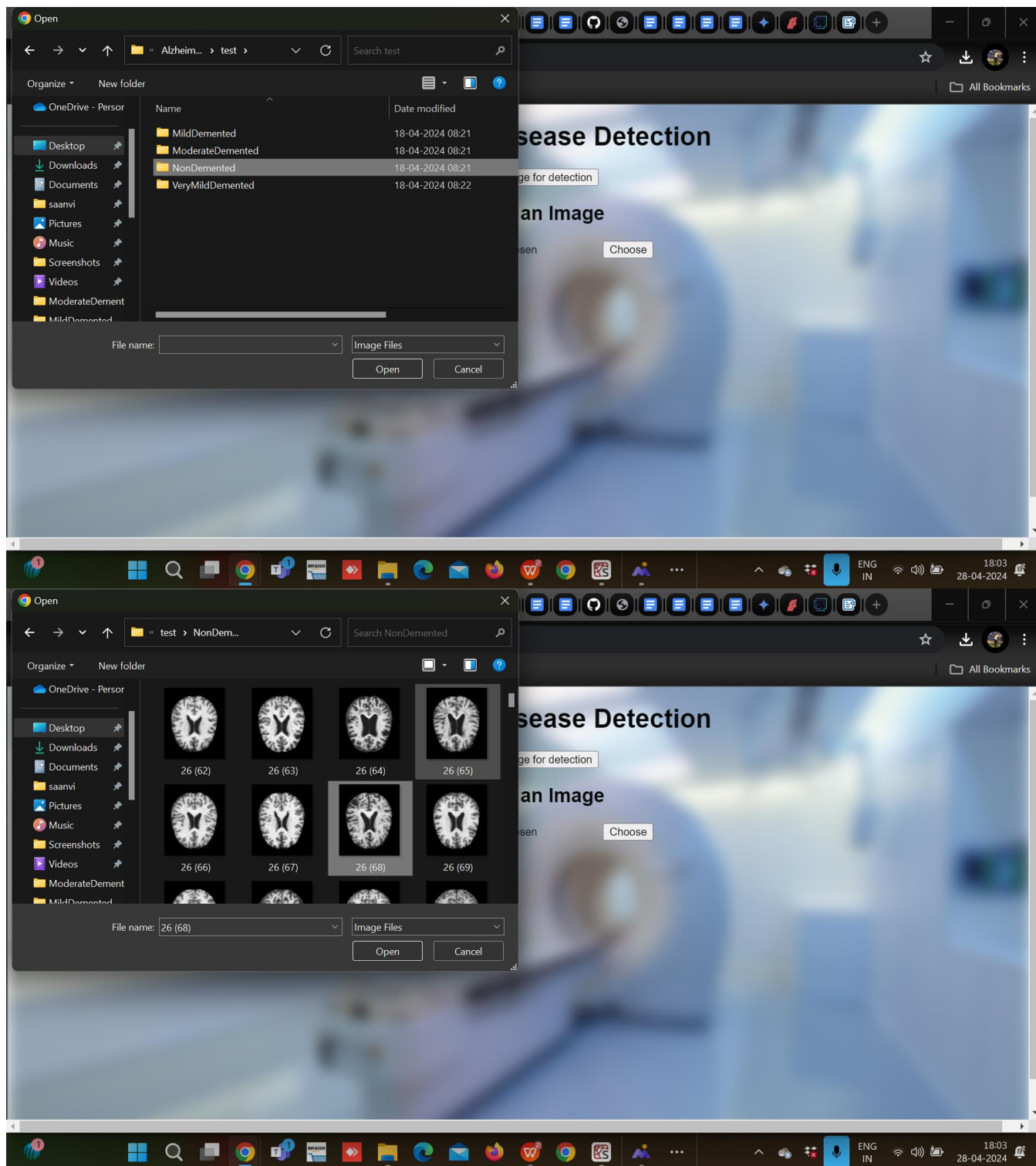


Second Page:-



If we click on the review your reports button it will redirect us to our testing page as shown below in the form of snippet.







7. Advantages & Disadvantages

Advantages:

High accuracy: Deep learning models like DenseNet169 can achieve very high accuracy in analyzing the highly complex 3D MRI data to detect subtle patterns and biomarkers associated with Alzheimer's pathology, even at early stages.

Automation: This approach can automatically process and evaluate MRI scans, saving significant time and effort compared to manual radiological reviews.

Objectivity: The deep learning model provides consistent, objective evaluations free

from human bias or fatigue.

Early detection: By identifying very mild Alzheimer's cases, it enables earlier diagnosis when treatments are more effective at slowing progression.

Interpretability: While complex, visualizations can provide insights into what image features the model uses to make predictions.

Scalability: The model can efficiently process very large datasets of MRI scans from multiple sites.

Disadvantages:

Data requirements: Training an accurate DenseNet169 model requires a very large, high-quality dataset of labeled 3D MRI scans across all stages of Alzheimer's.

Model complexity: DenseNet169 and other 3D convolutional networks are highly parameterized and computationally intensive to train.

Black box: Despite interpretability efforts, deep learning models like this are still often opaque "black boxes" to some degree.

Dataset biases: If the training data has biases in patient demographics or from certain scanners, it can reduce generalization.

Mistakes can propagate: Any errors made by the model in clinical practice can potentially have high costs if they lead to misdiagnosis.

Ethics concerns: There are ethical considerations around using AI systems for consequential medical diagnosis and patient triage.

8. Conclusion

DenseNet169 is a promising model for Alzheimer's classification, but it's not without limitations. Consider data quality, interpretability needs, computational resources, and explore alternative models to determine the best approach for your specific project.

9. Future Scope

SCOPE OF THE PROJECT: Recent advances in early detection and algorithmic AD categorization have produced extensive multimodal neuroimaging data. MRI imaging and the findings of genetic sequencing are two different methods for AD research. Making a decision requires careful consideration of several modalities, which takes time. Patients may also experience radioactive effects when viewing modalities like MRI pictures. In this study, we hypothesize that the MRI mode gains from its superior tissue contrast, increased imaging elasticity, absence of ionizing radiation, and capacity to deliver valuable data on human brain architecture. Improved computer-aided diagnostic tools are needed to analyze MRI images and identify whether individuals have Alzheimer's disease or are well. Conventional deep learning algorithms conduct AD cataloging on unprocessed MRI images using the cortical surface as a parameter to the CNN

10. Appendix

10.1. Source Code

Gmail YouTube Maps

alzheimer-s-classification-densenet169.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] !unzip '/content/archive (11).zip'
```

```
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem335.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem336.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem337.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem338.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem339.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem340.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem341.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem342.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem343.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem344.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem345.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem346.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem347.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem348.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem349.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem350.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem351.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem352.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem353.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem354.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem355.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem356.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem357.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem358.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem359.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem360.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem361.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem362.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem363.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem364.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem365.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem366.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem367.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem368.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem369.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem370.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem371.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem372.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem373.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem374.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem375.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem376.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem377.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem378.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem379.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem380.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem381.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem382.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem439.jpg
inflating: Alzheimer_s Dataset/train/MildDemented/mildDem440.jpg
```

```
train_data_dir = '/content/Alzheimer_s Dataset/train'
test_data_dir = '/content/Alzheimer_s Dataset/test'
```

Start coding or generate with AI.

IMPORT LIBRARIES

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

from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

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

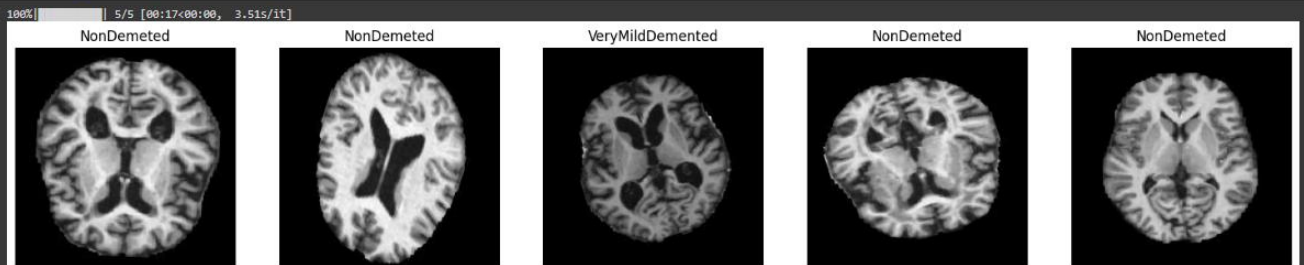
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout, Flatten, Dense, Activation, MaxPool2D, Conv2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint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

DATA AUGMENTATION

```
Found 4098 images belonging to 4 classes.
```

```
Found 1023 images belonging to 4 classes.
```

```
for i in tqdm(range(95)):
    r1 = np.random.randint(len(train_dataset))
    rand2 = np.random.randint(100)
    ax[i].imshow(train_dataset[rand1][0][rand2])
    ax[i].axis('off')
    a = train_dataset[rand1][1][rand2]
    if a[0] == 1:
        ax[i].set_title('MildDemented')
    elif a[1] == 1:
        ax[i].set_title('ModerateDemented')
    elif a[2] == 1:
        ax[i].set_title('NonDemented')
    elif a[3] == 1:
        ax[i].set_title('VeryMildDemented')
```



MODEL BUILDING

```
[ ] # Model Initialization
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet169\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5  
51877672/51877672 [=====] - 1s 0us/step
```

```
[ ] # Freezing Layers
```

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

```
[ ] # Building Model
```

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

Summary

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
densenet169 (Functional)	(None, 7, 7, 1664)	12642880
dropout (Dropout)	(None, 7, 7, 1664)	0
flatten (Flatten)	(None, 81536)	0
batch_normalization (Batch Normalization)	(None, 81536)	326144
dense (Dense)	(None, 2048)	166987776
batch_normalization_1 (Batch Normalization)	(None, 2048)	8192
activation (Activation)	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_1 (Dense)	(None, 1024)	2098176
batch_normalization_2 (Batch Normalization)	(None, 1024)	4096
activation_1 (Activation)	(None, 1024)	0
dropout_2 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 4)	4100

```
=====
Total params: 182071364 (694.55 MB)
Trainable params: 169259268 (645.67 MB)
Non-trainable params: 12812096 (48.87 MB)
```

```
Non-trainable params: 12812096 (48.87 MB)
```

[] # Model Compile

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

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

WARNING: `absl.lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g., `tf.keras.optimizers.legacy.Adam`.

Defining Callbacks

```
filepath = './best_weights.hdf5'

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

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

callback_list = [earlystopping, checkpoint]
```

```
model_history=model.fit(train_dataset,
                        validation_data=valid_dataset,
                        epochs = 10,
                        callbacks = callback_list,
                        verbose = 1)
```

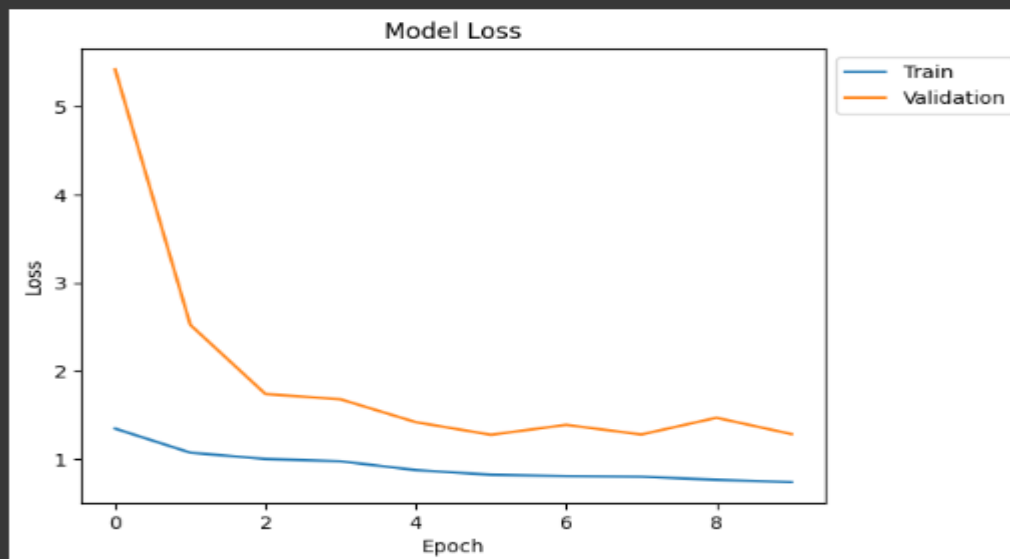
Epoch 1/10
33/33 [=====] - ETA: 0s - loss: 1.3450 - auc: 0.7910
Epoch 1: val_auc improved from -inf to 0.69909, saving model to ./best_weights.hdfs
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via 'model.save()'. This file format is considered legacy. We recommend using instead the native saving API 'save_model()'.

33/33 [=====] - 1373s 41s/step - loss: 1.3450 - auc: 0.7910 - val_loss: 5.4237 - val_auc: 0.6991
Epoch 2/10
33/33 [=====] - ETA: 0s - loss: 1.0718 - auc: 0.8356
Epoch 2: val_auc improved from 0.69909 to 0.71963, saving model to ./best_weights.hdfs
33/33 [=====] - 1364s 41s/step - loss: 1.0718 - auc: 0.8356 - val_loss: 2.5231 - val_auc: 0.7196
Epoch 3/10
33/33 [=====] - ETA: 0s - loss: 1.0003 - auc: 0.8462
Epoch 3: val_auc improved from 0.71963 to 0.76162, saving model to ./best_weights.hdfs
33/33 [=====] - 1359s 41s/step - loss: 1.0003 - auc: 0.8462 - val_loss: 1.7376 - val_auc: 0.7616
Epoch 4/10
33/33 [=====] - ETA: 0s - loss: 0.9738 - auc: 0.8524
Epoch 4: val_auc improved from 0.76162 to 0.76515, saving model to ./best_weights.hdfs
33/33 [=====] - 1376s 42s/step - loss: 0.9738 - auc: 0.8524 - val_loss: 1.6775 - val_auc: 0.7651
Epoch 5/10
33/33 [=====] - ETA: 0s - loss: 0.8739 - auc: 0.8730
Epoch 5: val_auc did not improve from 0.76515
33/33 [=====] - 1367s 42s/step - loss: 0.8739 - auc: 0.8730 - val_loss: 1.4180 - val_auc: 0.7637
Epoch 6/10
33/33 [=====] - ETA: 0s - loss: 0.8207 - auc: 0.8847
Epoch 6: val_auc improved from 0.76515 to 0.79470, saving model to ./best_weights.hdfs
33/33 [=====] - 1399s 42s/step - loss: 0.8207 - auc: 0.8847 - val_loss: 1.2748 - val_auc: 0.7947
Epoch 7/10
33/33 [=====] - ETA: 0s - loss: 0.8056 - auc: 0.8893
Epoch 7: val_auc did not improve from 0.79470
33/33 [=====] - 1350s 41s/step - loss: 0.8056 - auc: 0.8893 - val_loss: 1.3872 - val_auc: 0.7826
Epoch 8/10
33/33 [=====] - ETA: 0s - loss: 0.7991 - auc: 0.8893
Epoch 8: val_auc did not improve from 0.79470
33/33 [=====] - 1363s 41s/step - loss: 0.7991 - auc: 0.8893 - val_loss: 1.2780 - val_auc: 0.7787
Epoch 9/10
33/33 [=====] - ETA: 0s - loss: 0.7631 - auc: 0.8982
Epoch 9: val_auc did not improve from 0.79470
33/33 [=====] - 1348s 41s/step - loss: 0.7631 - auc: 0.8982 - val_loss: 1.4680 - val_auc: 0.7624
Epoch 10/10
33/33 [=====] - ETA: 0s - loss: 0.7372 - auc: 0.9014
Epoch 10: val_auc improved from 0.79470 to 0.79961, saving model to ./best_weights.hdfs
33/33 [=====] - 1435s 44s/step - loss: 0.7372 - auc: 0.9014 - val_loss: 1.2822 - val_auc: 0.7996

MODEL EVALUATION

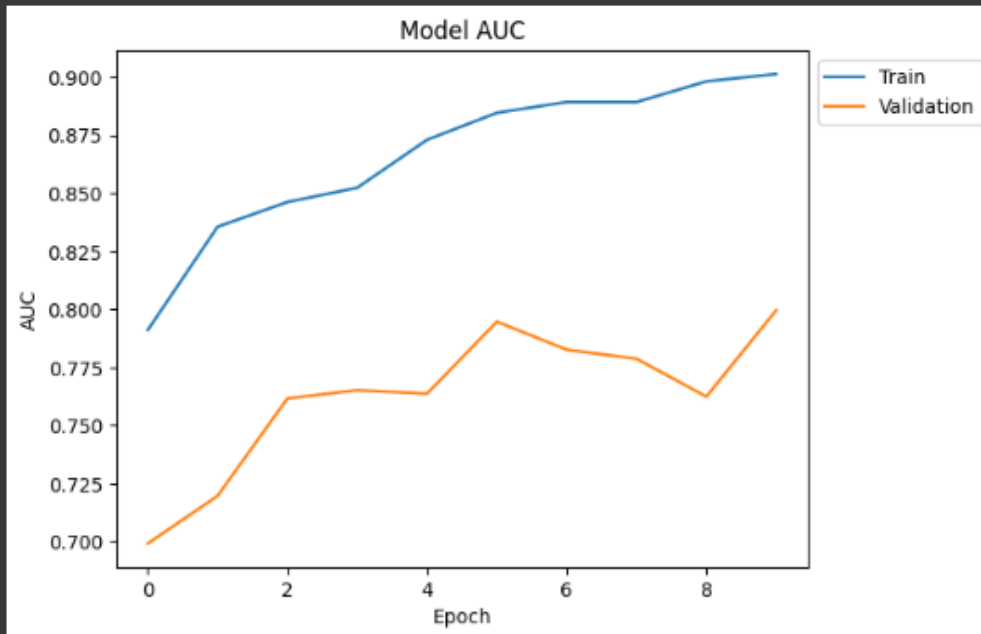
```
[ ] # Summarize history for loss
```

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



```
# Summarize history for loss
```

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



```
# Test Data
```

```
# Test Data
```

```
test_dataset = test_datagen.flow_from_directory(directory = '/content/Alzheimer's Dataset/test',
                                                target_size = (224,224),
                                                class_mode = 'categorical',
                                                batch_size = 128)
```

```
Found 1279 images belonging to 4 classes.
```

```
# Evaluating Loss and AUC
```

```
model.evaluate(test_dataset)
```

```
18/18 [=====] - 297s 29s/step - loss: 1.0717 - auc: 0.8530
[1.0716747045516968, 0.852987335285070]
```

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

```
model.save("model.h5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3183: UserWarning: You are saving your model as an H5 file via 'model.save()'. This file format is considered legacy. We recommend using instead the native Keras format, e.g., 'model.save('my_model.keras')'.
  saving_api.save_model(
```

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

```
model = load_model('/content/model.h5')
```

```
answer = model.predict(img)
```

```
1/1 [=====] - 4s 4s/step
```

```
class_label = np.argmax(answer)
```

```
class_name = id[class_label]
```

```
Code | Text
# Test Case 1: Non-Dementia

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

img = load_img('/content/Alzheimer_s Dataset/test/NonDemented/26 (69).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)

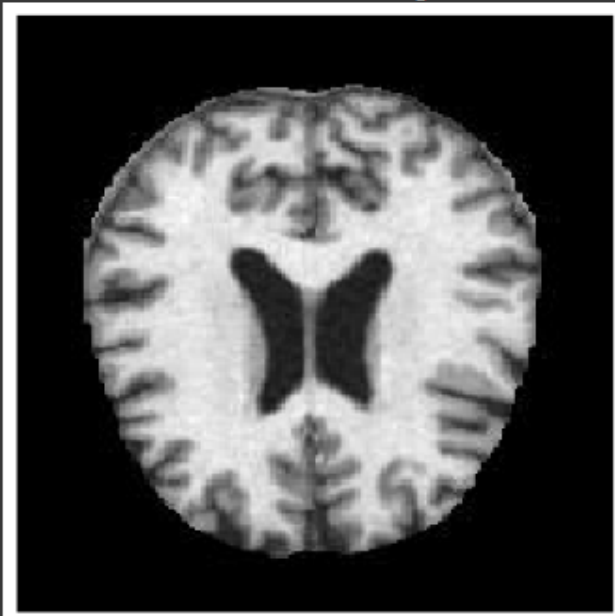
# Load the model
model = load_model('/content/model.h5')

# Predict the class
answer = model.predict(img)

# Extract the class label and name
class_label = np.argmax(answer)
class_name = idc[class_label]

# Calculate and print the probability
probability = round(np.max(answer)*100,2)
print(probability, '% chances are there that the image is', class_name)
```

1/1 [=====] - 4s 4s/step
48.12 % chances are there that the image is NonDemented



```
# Test Case 2: MildDemented

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

img = load_img('/content/Alzheimer_s Dataset/test/MildDemented/26 (23).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)

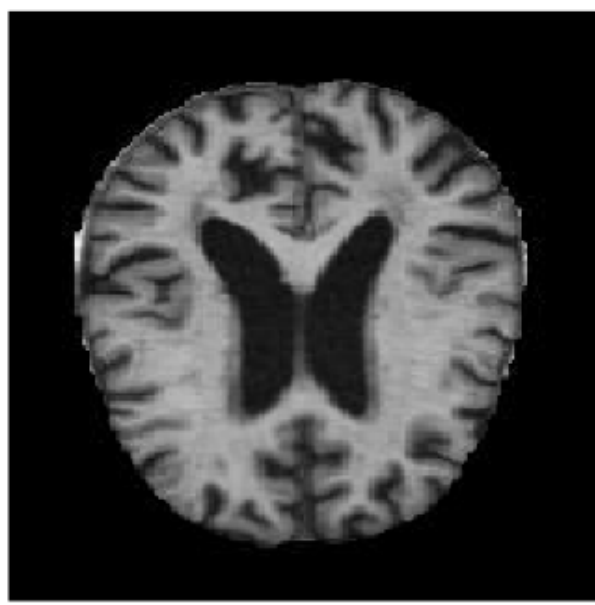
# Load the model
model = load_model('/content/model.h5')

# Predict the class
answer = model.predict(img)

# Extract the class label and name
class_label = np.argmax(answer)
class_name = idc[class_label]

# Calculate and print the probability
probability = round(np.max(answer)*100,2)
print(probability, '% chances are there that the image is', class_name)
```

```
1/1 [=====] - 5s 5s/step
69.94 % chances are there that the image is MildDemented
```



```
+ Code + Text

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

img = load_img('/content/Alzheimer_s_Dataset/test/NonDemented/26 (49).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)


# Load the model
model = load_model('/content/model.h5')

# Predict the class
answer = model.predict(img)

# Extract the class label and name
class_label = np.argmax(answer)
class_name = idc[class_label]

# Calculate and print the probability
probability = round(np.max(answer)*100,2)
print(probability, '% chances are there that the image is', class_name)

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7ebcb29b37f0> triggered tf.funct
1/1 [=====] - 4s 4s/step
86.88 % chances are there that the image is NonDemented
```



10.2.GitHub & Project Demo Link

GitHub Link:-[Click here](#)

Project Demo Link:-[Click Here](#)