**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6. SYSTEM IMPLEMENTATION**

**6.1 MODULE DESIGN SPECIFICATION**

The model for real-time image detection using CNN in Python involves several key steps:

* **Data collection and acquisition**
* **Data pre-processing**
* **Feature extraction**
* **Classification model**
* **Model evaluation**
* **Model deployment**

**6.1.1 Data collection and acquisition:**

The dataset used for training the CNN model should be representative of the image that need to be detected. The dataset can be collected from various sources, such as public datasets or custom data collection efforts. Data used in the preparation of this report were obtained from the publicly available Alzheimer’s Disease Neuroimaging Initiative (ADNI) database and the Open Access Series of Imaging Studies (OASIS) project database. The most recent visit in which a diagnosis was made was considered the best available “ground-truth” to train the classifiers.

**6.1.2 Data pre-processing:**

This step involves pre-processing of the MRI images such as skull stripping, intensity normalization, and image registration. It is important to ensure that the images are of the same size, orientation, and spatial resolution to facilitate easy comparison and analysis.

**6.1.3 Feature extraction:**

This step involves extracting features from the pre-processed images, such as gray matter volume, cortical thickness, and hippocampal shape. These features are then used to classify the images into Alzheimer's disease (AD) like Mild Demented, Moderate Demented, Non Demented and Very Mild Demented.

**6.1.4 Classification model:**

The extracted features are then used to train a classification model, as convolutional neural network (CNN), to classify the images as Mild Demented, Moderate Demented, Non Demented and Very Mild Demented.

**6.1.5 Model evaluation**:

The trained model is evaluated using various metrics such as accuracy, sensitivity, specificity, and F1 score. The evaluation process helps to determine the effectiveness of the classification model and identify areas for improvement.

**6.1.6 Model deployment:**

The final step involves deploying the classification model to classify new MRI images into Mild Demented, Moderate Demented, Non Demented and Very Mild Demented. The model can be deployed as a standalone application, integrated into existing healthcare systems, or used as a diagnostic tool by healthcare professionals.

**CHAPTER 7**

**SYSTEM TESTING**

### 7. SYSTEM TESTING

#### 7.1 SOFTWARE TESTING

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists.

##### 7.1.1 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.

##### 7.1.2 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 centred 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, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

##### 7.1.3 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. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

##### 7.1.4 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

##### 7.1.5 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works

##### 7.1.6 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points

##### 7.1.7 OUTPUT TESTING

* Output of test cases compared with the expected results created during design of test cases.
* Asking the user about the format required by them tests the output generated or displayed by the system under consideration.
* Here, the output format is considered into two was, one is on screen and another one is printed format.
* The output on the screen is found to be correct as the format was designed in the system design phase according to user needs.
* The output comes out as the specified requirements as the user’s hard copy.

##### 7.1.8 USER ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

* Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data.
* Two set of acceptance test to be run: 1. Those developed by quality assurance group 2. Those developed by customer

**7.2 TEST CASES**

### TEST REPORT: 01

**PRODUCT:** AUTOMATED DETECTION OF ALZHEIMER DISEASE

**USE CASE:** UPLOAD IMAGE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST**  **CASE**  **ID** | **TEST CASE /**  **ACTION TO BE**  **PERFORMED** | **EXPECTED RESULT** | **ACTUAL**  **RESULT** | **PASS / FAIL** |
| 1 | Upload the image as an input | Uploaded | As expected | PASS |

**Table-7.2.1** TEST CASE FOR IMAGE UPLOAD

### TEST REPORT: 02

**PRODUCT:** AUTOMATED DETECTION OF ALZHEIMER DISEASE

**USE CASE:** IDENTIFY AND DETECT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST**  **CASE**  **ID** | **TEST CASE /**  **ACTION TO BE**  **PERFORMED** | **EXPECTED RESULT** | **ACTUAL**  **RESULT** | **PASS / FAIL** |
| 1 | Identify the intensity from the MRI image | Identified  Successfully | As expected | PASS |
| 2 | Show the detected  intensity from the image | Detected  Successfully | As expected | PASS |

**Table-7.2.2** TEST CASE FOR IDETIFY AND DETECT

**CHAPTER 8**

**CONCLUSION &**

**FUTURE ENHANCEMENT**

### 8. CONCLUSION AND FUTURE ENHANCEMENTS

#### 8.1 CONCLUSION

As a result, we put our theory of transfer learning from CNN to other classifiers through its paces, following the steps in sequential order. It is possible for us to draw one of the defining qualities for training the classifier is the ability to transfer learnt parameters and features from a CNN that has been trained. If the feature transformation, selection, and classification processes are carried out in an intelligent manner, the CNN features trained classifier can achieve a higher level of performance than CNN networks itself. The performance of CNN can even be enhanced by properly modifying and tuning the architecture of CNN, as well as by appropriately optimizing the classification system.

**8.2 FUTURE ENHANCEMENTS**

The use of convolutional neural network (CNN) algorithm has shown promising results in the prediction of Alzheimer's disease. As technology advances, there is a great potential for further enhancement in this area. One possible future direction is to incorporate multimodal data, such as imaging, genetic, and clinical data, to improve the accuracy of prediction. Additionally, transfer learning can be employed to improve the generalizability of the model by leveraging pre-trained models on related tasks. Furthermore, the use of explainable AI techniques can help to interpret the model's predictions and provide insights into the underlying biological mechanisms of the disease. Overall, the future of prediction of Alzheimer's disease using CNN algorithm is bright, and further research and development in this area can have a significant impact on early detection and treatment of the disease.

**APPENDICES**

### APPENDICES

### A1. CODING

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.color import rgb2gray

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

train\_datagen = ImageDataGenerator(rescale = 1./255,

rotation\_range=30,

zoom\_range=0.2,

horizontal\_flip=True,

vertical\_flip=True,

validation\_split = 0.2)

valid\_datagen = ImageDataGenerator(rescale = 1./255,

validation\_split = 0.2)

test\_datagen = ImageDataGenerator(rescale = 1./255)

train\_dataset = train\_datagen.flow\_from\_directory(directory = 'D:/brain alzheimer detection/Dataset',

target\_size = (224,224),

class\_mode = 'categorical',

subset = 'training',

batch\_size = 128)

valid\_dataset = valid\_datagen.flow\_from\_directory(directory = 'D:/brain alzheimer detection/Dataset',

target\_size = (224,224),

class\_mode = 'categorical',

subset = 'validation',

batch\_size = 128)

fig, ax = plt.subplots(nrows = 1, ncols = 5, figsize=(20,20))

for i in tqdm(range(0,5)):

rand1 = 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('Mild Dementia')

elif a[1] == 1:

ax[i].set\_title('Moderate Dementia')

elif a[2] == 1:

ax[i].set\_title('Non Demetia')

elif a[3] == 1:

ax[i].set\_title('Very Mild Dementia')

# Model Initialization

base\_model = DenseNet169(input\_shape=(224,224,3),

include\_top=False,

weights="imagenet")

# 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 Compile

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

model.compile(loss='categorical\_crossentropy',

metrics=[tensorflow.keras.metrics.AUC(name = 'auc')],

optimizer=OPT)

# 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 =30 ,

callbacks = callback\_list,

verbose = 1)

# 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\_dataset = test\_datagen.flow\_from\_directory(directory = 'D:/brain alzheimer detection/Dataset',

target\_size = (224,224),

class\_mode = 'categorical',

batch\_size = 128)

# Evaluating Loss and AUC

model.evaluate(test\_dataset)

# Test Case 1: Non-Dementia

dic = test\_dataset.class\_indices

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

img = load\_img('D:/brain alzheimer detection/Dataset/Moderate\_Demented/moderate\_9.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)

answer = np.argmax(model.predict(img),axis=1)

probability = round(np.max(model.predict(img)\*100),2)

print(probability, '% chances are there that the image is',idc[answer[0]])

### A.2 SAMPLE SCREEN

In fig A.2.1 Open Anaconda Prompt Console and activate brain. Change the directory to the location where the codes are stored. Call the Jupyter Notebook from here.

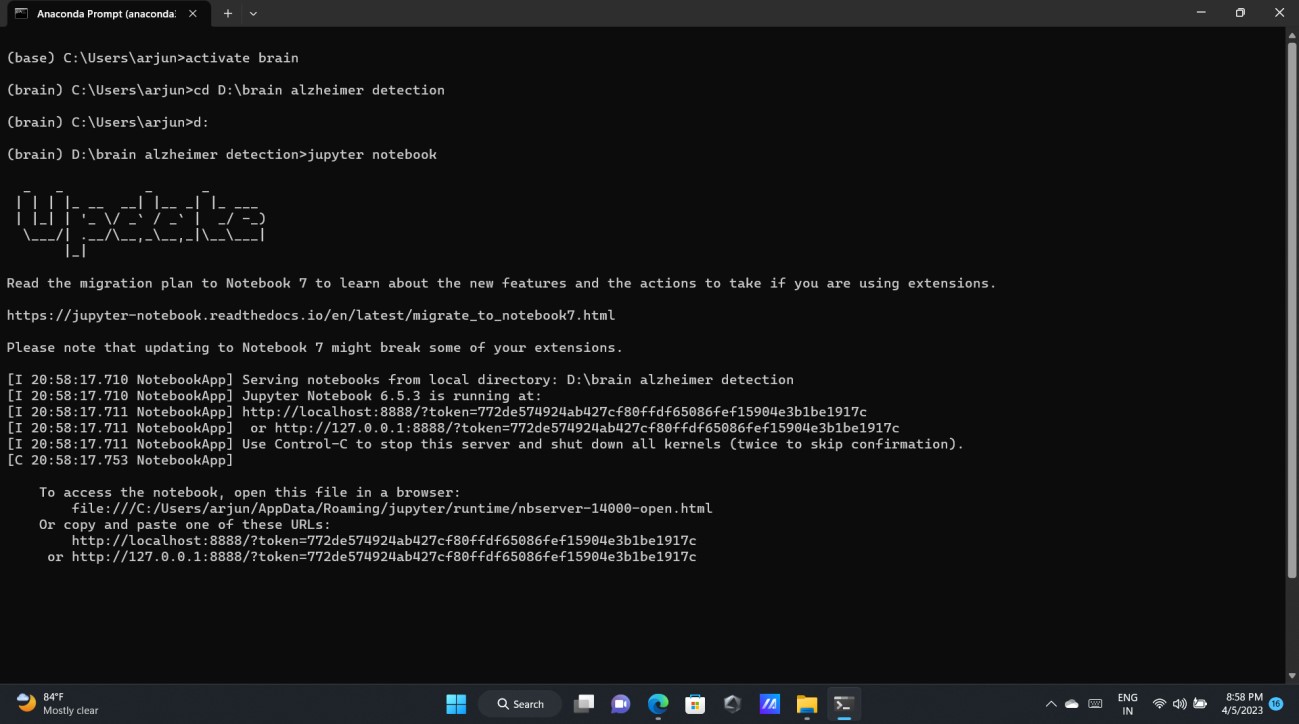


Fig A.2.1 Anaconda Prompt Console

In fig A.2.2 This is the UI of the Jupyter Notebook , Here we can see all our project folders.

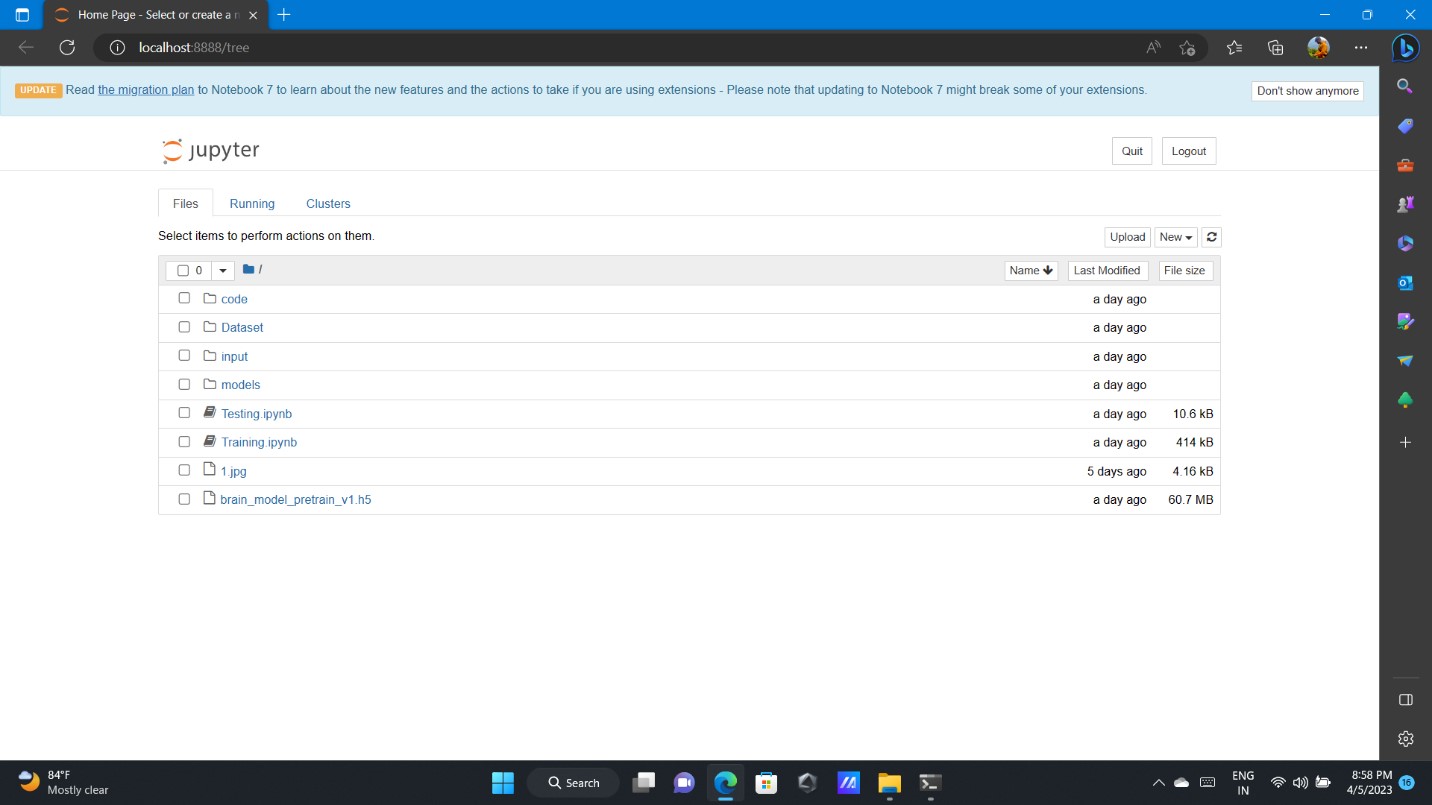
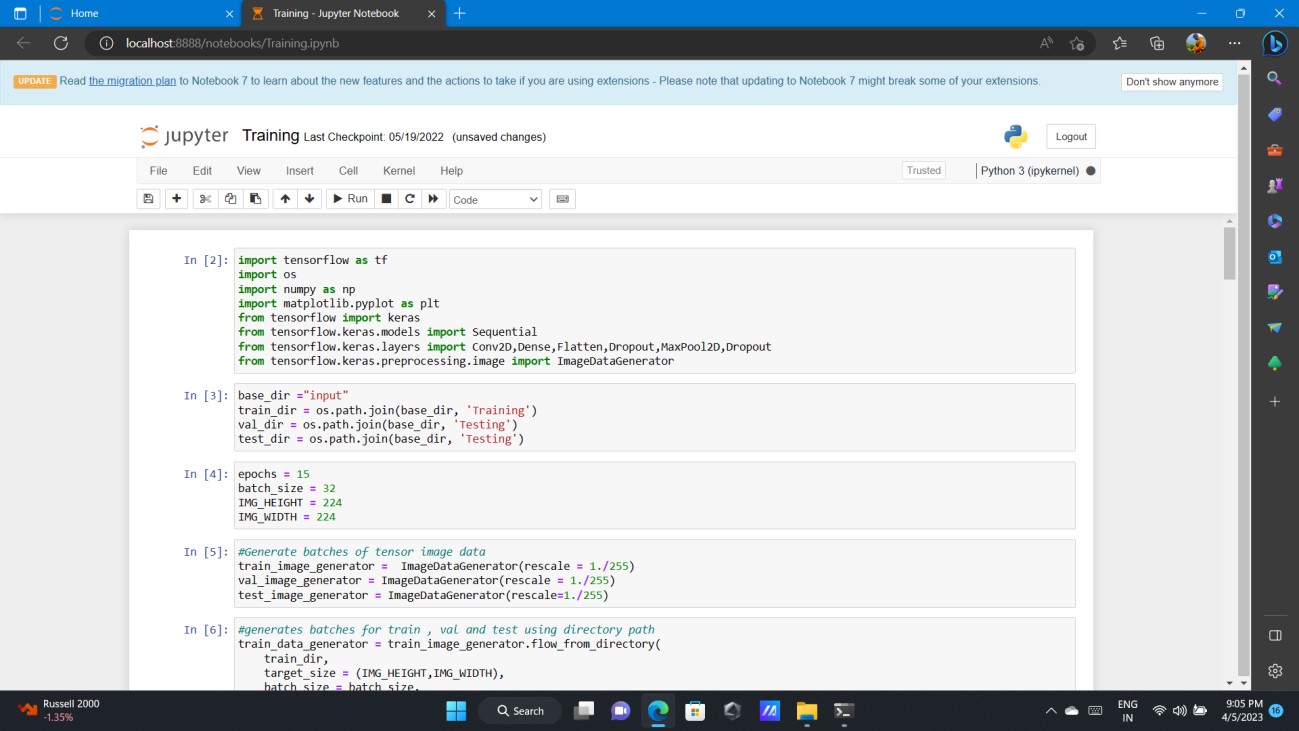
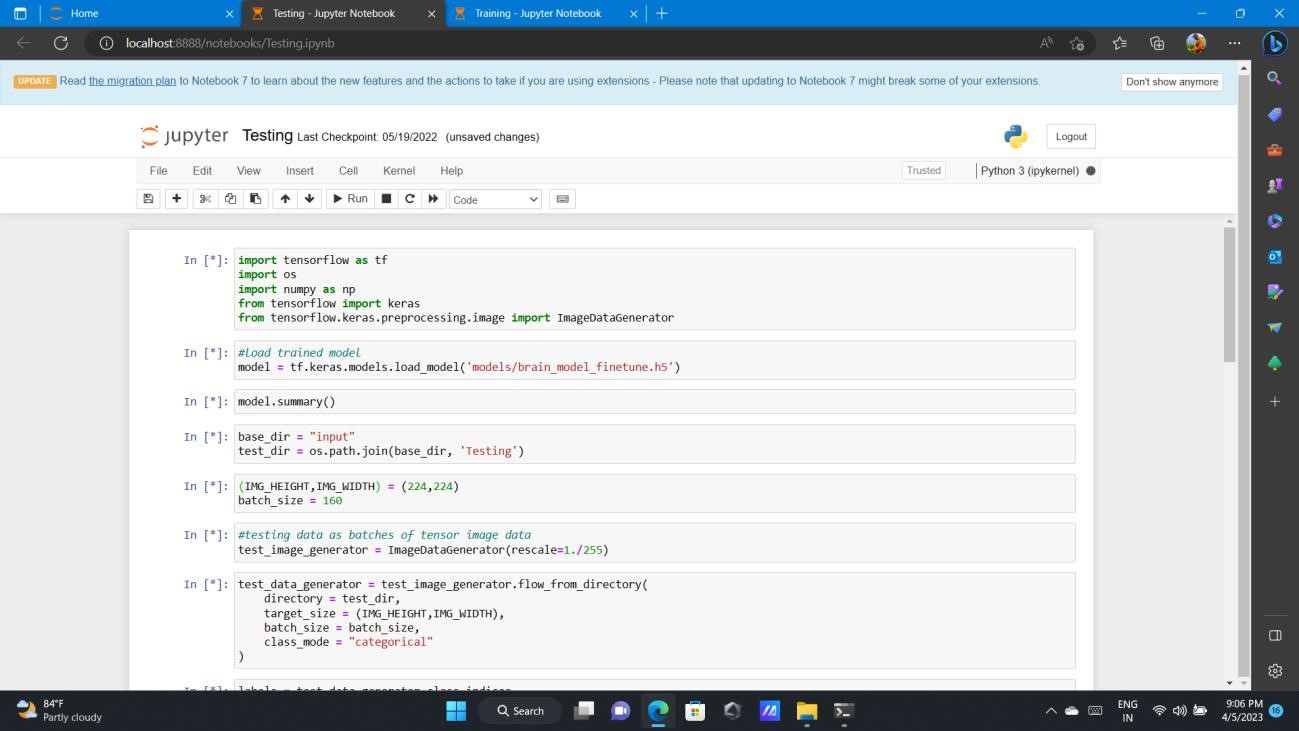


Fig A.2.2 UI of the Jupyter Notebook

In fig A.2.3 Run all the cells in the Training folder (module)

Fig A.2.3 Training folder

In fig A.2.4 Run all the cells in the Testing folder (module)

Fig A.2.4 Testing Folder

In fig A.2.5 Now click on the code folder and select the Main Source File.

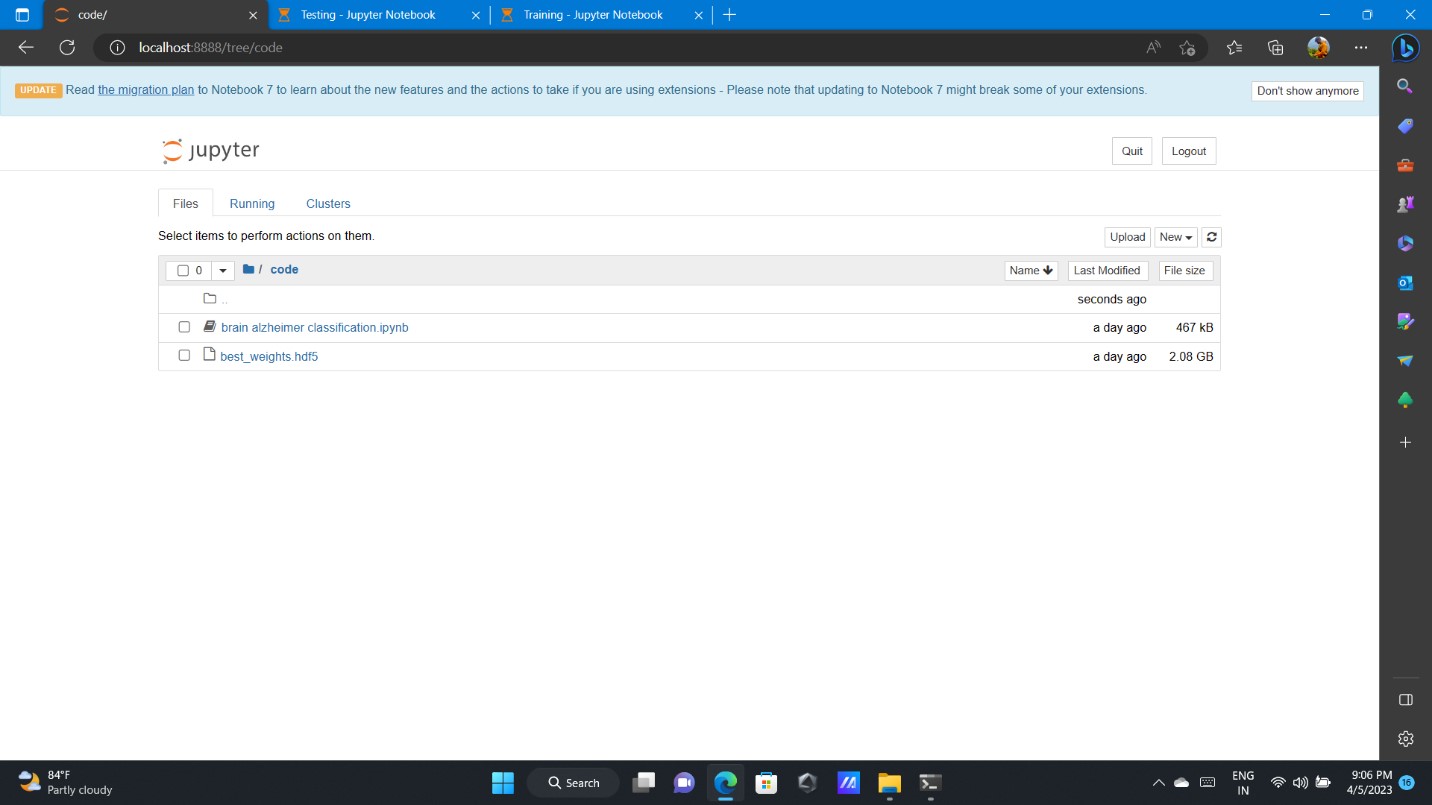
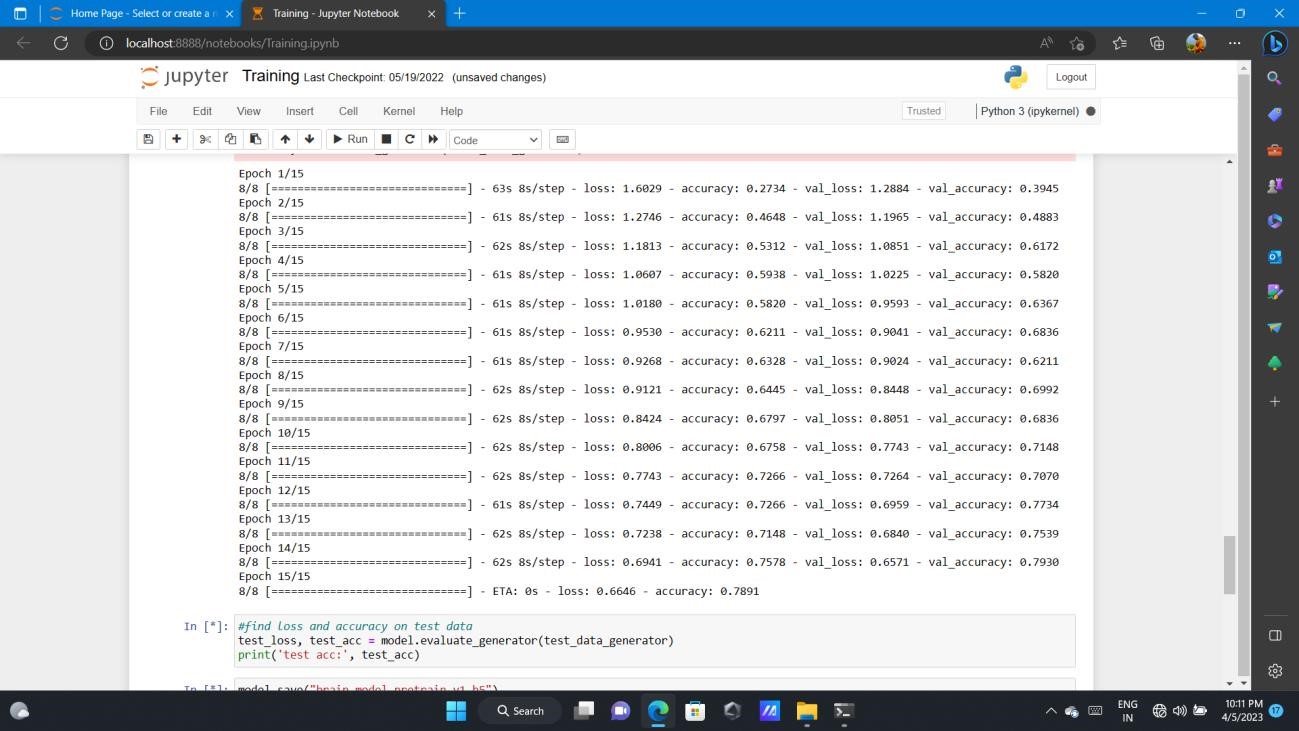
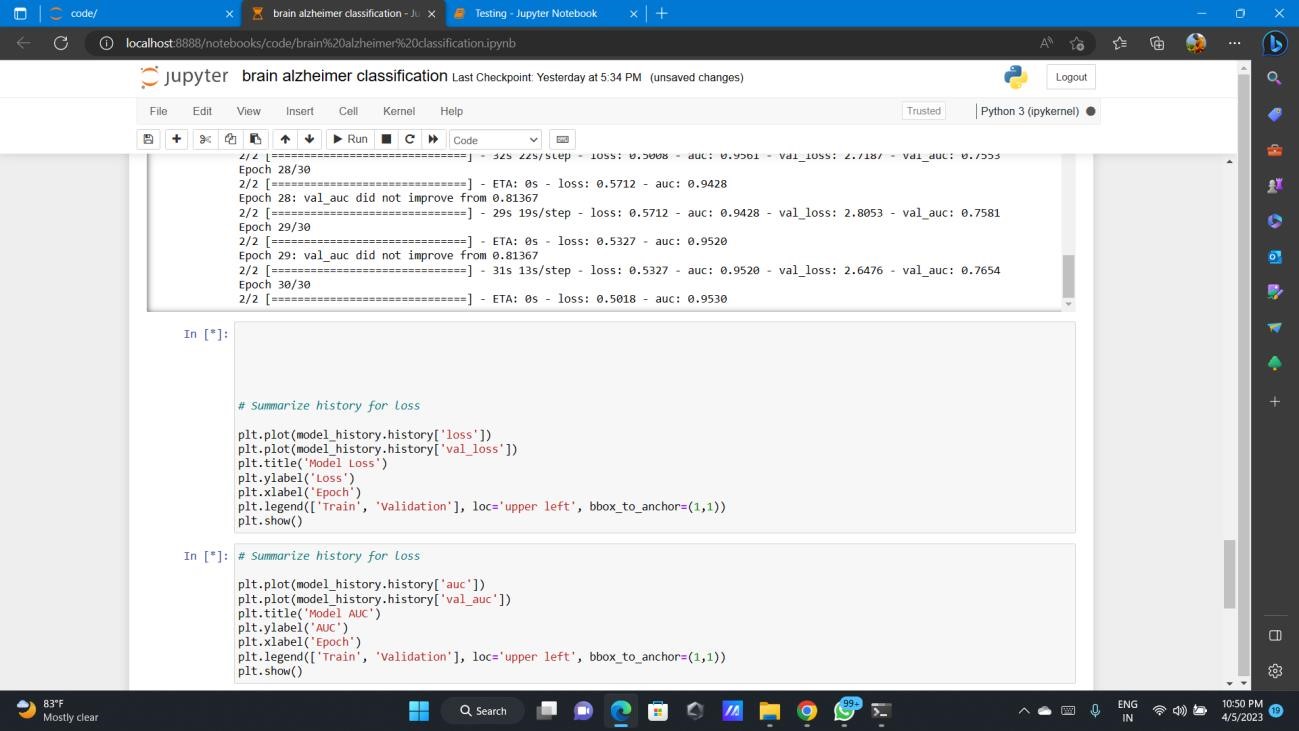


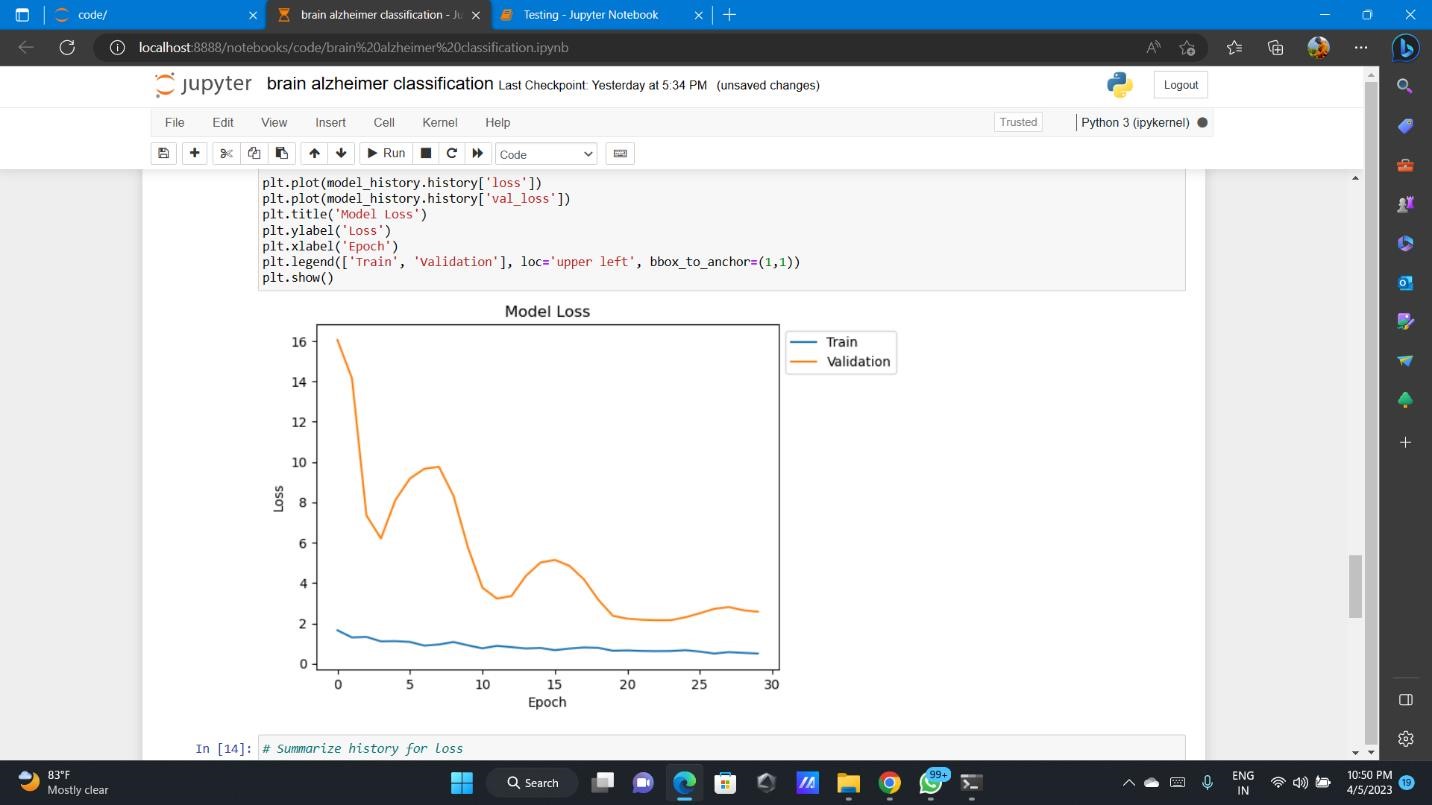
Fig A.2.5 Main Source File

In fig A.2.6 & A.2.7 This is how all the Epoches are completed after a successful run

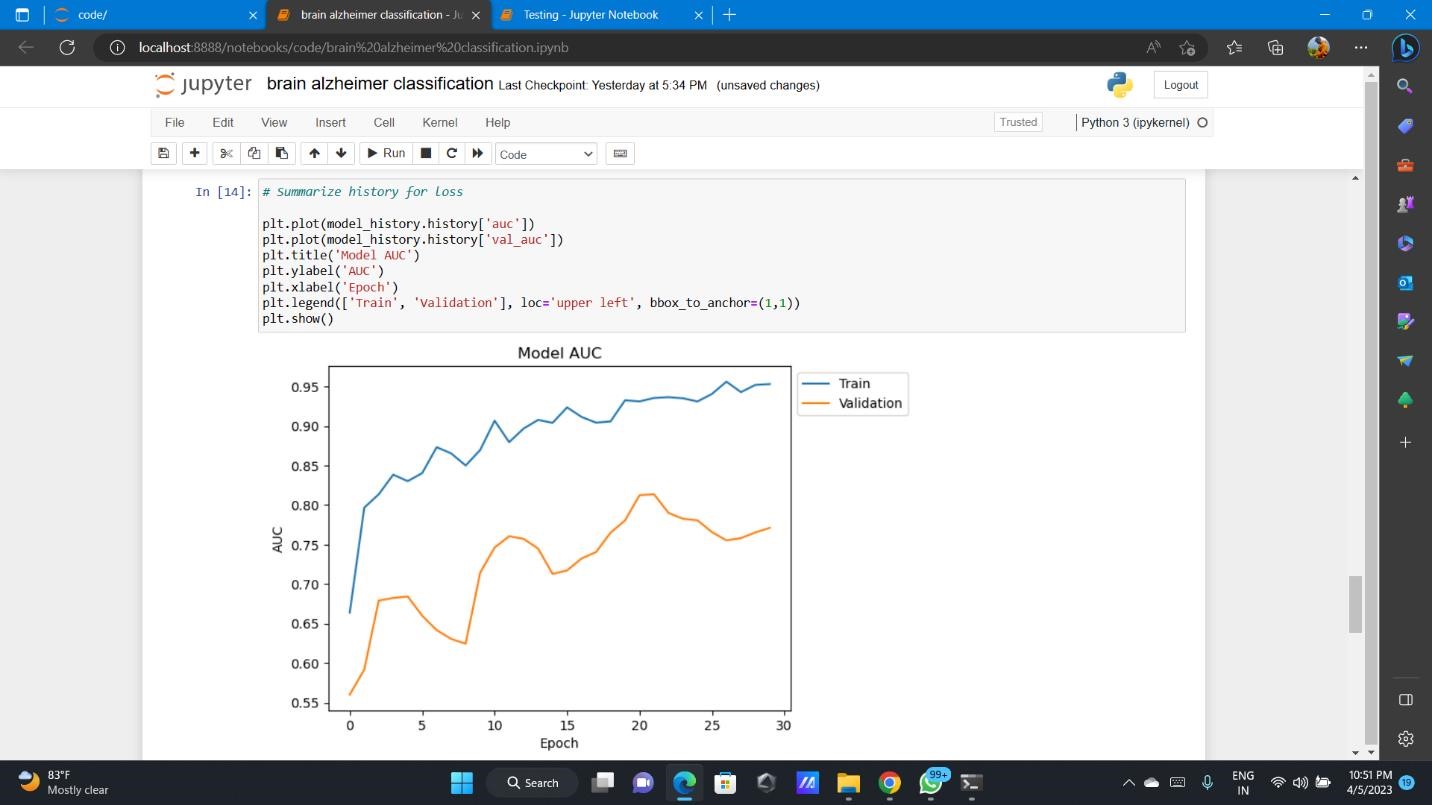
Fig A.2.6 Epoches

Fig A.2.7 Epoches

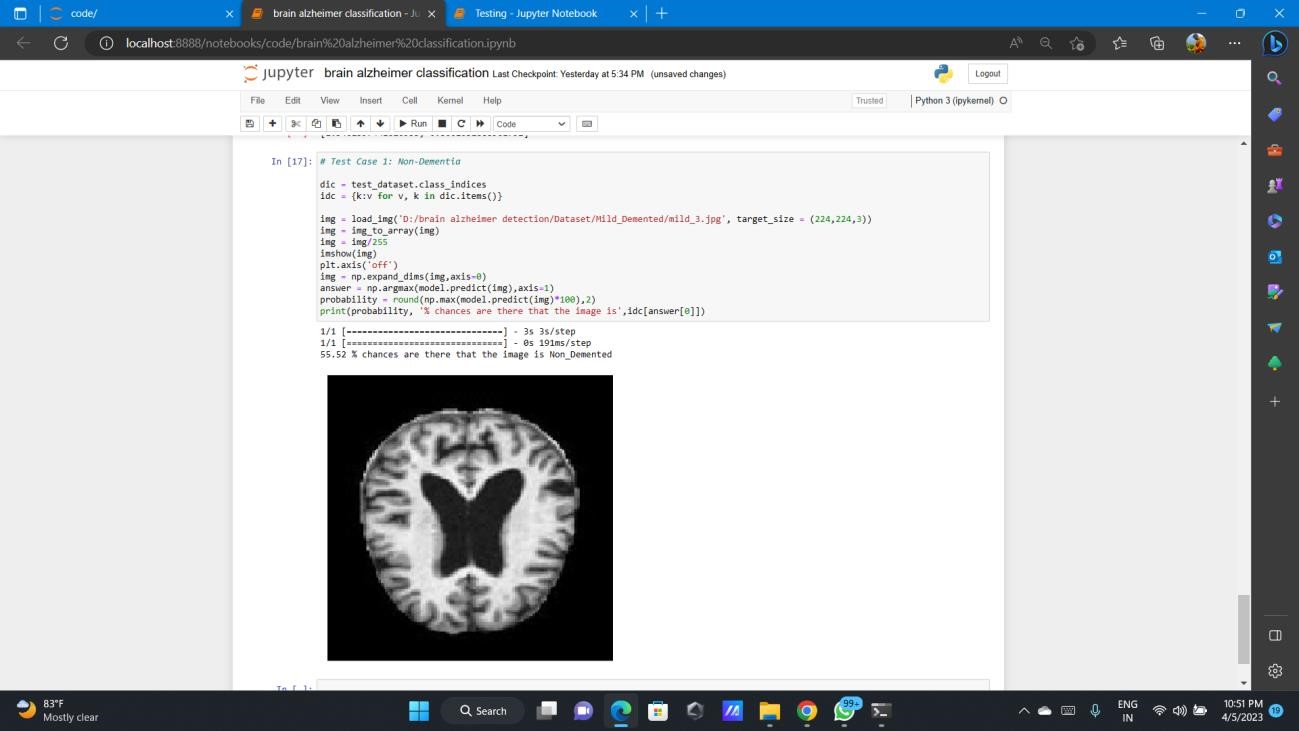
In fig A.2.8 The Train and Validation graph is successfully plotted and shown below (Model Loss)

Fig A.2.8 Validation Graph (Model Loss)

In fig A.2.9 The Train and Validation graph is successfully plotted and shown below (Model Accuracy)

Fig A.2.9 Validation Graph (Model Accuracy)

In fig A.2.10 The final successful input and output are shown below

Fig A.2.10 Final Input and Output

**REFERENCES**

**REFERENCES**

1. Y. Zhang, S. Wang, and Z. Dong, “Classification of Alzheimer disease based on structural magnetic resonance imaging by kernel support vector machine decision tree”, Progress In Electromagnetics Research, vol. 144, January 2014.
2. S. Chaplot, L. M. Patnaik, and N. R. Jagannathan, “Classification of magnetic resonance brain images using wavelets as input to support vector machine and neural network”, Biomedical Signal Processing and Control, vol. 1, pp. 86-92, January 2006.
3. M. Maitra, and A. Chatterjee, “A Slantlet transform based intelligent system for magnetic resonance brain image classification”, Biomedical Signal Processing and Control, vol. 1, pp. 299-306, October 2006.
4. E. S. A. El-Dahshan, T. Hosny, and A. B. M. Salem, “Hybrid intelligent techniques for

MRI brain images classification”, Digital Signal Processing, vol. 1, pp. 299-306, October 2006.

1. Y. Zhang, L. Wu, and S. Wang, “Magnetic resonance brain image classification by an improved artificial bee colony algorithm”, Progress In Electromagnetics Research, vol. 116, pp. 65-79, 2011.
2. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks”, Advances in neural information processing systems, pp. 1097- 1105, 2012.
3. K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition” arXiv:1512.03385, 2015.
4. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions”, In Proceedings of the IEEE conference on computer vision and pattern recognition pp. 1-9, 2015.
5. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587, 2014.
6. N. Tajbakhsh, J. Shin, S. Gurudu, C. Kendall, M. Gotway, and J. Liang, “Convolutional

Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?”, IEEE Transactions on Medical Imaging, vol. 35, pp. 1299-1312, May 2016.

1. H. Shin, H. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, and J. Yao, “Deep Convolutional

Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics, and Transfer Learning”, IEEE Transactions on Medical Imaging, vol. 35, pp. 1285-1298, May 2016.

1. H. R. Roth, L. Lu, J. Liu, J. Yao, A.Seff, K. Cherry, L. Kim, and Ronald M. Summers, “Improving Computer-Aided Detection Using Convolutional Neural Networks and

Random View Aggregation”, IEEE Transactions on Medical Imaging, vol. 35, pp. 11701181, May 2016.

1. A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf:
2. An astounding baseline for recognition”, Proc.IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, pp. 512-519, 2014.