Journal of Information Technology and Digital World (ISSN: 2582-418X)
www.irojournals.com/itdw/

# Real-Time Images and Deep Learning to Identify Alzheimer's Disease in a Web Application

**V. Surendar[1], E. Sriram[2], K. Karunakaran[3], S. Sudharshan[4], M. Yuvaraja[5]**

Department of Information Technology, Hindustan Institute of Technology and Science, Chennai. Tamil Nadu, India.

**E-mail:** [1]20134005@student.hindustanuniv.ac.in, [2]20134033@student.hindustanuniv.ac.in, [3]20134013@student.hindustanuniv.ac.in, [4]20134010@student.hindustanuniv.ac.in, [5]20134045@student.hindustanuniv.ac.in

## Abstract

Alzheimer's is a neurological condition that affects millions of individuals throughout the world. Early detection is critical for optimal treatment and management of this condition. This research presents a classification model that uses Convolutional Neural Networks (CNNs) to reliably identify the Alzheimer's disease using brain MRI scans. A publicly available dataset of brain MRI images divided into four categories: mild dementia, moderate dementia, non-dementia, and very mild dementia, has been utilized. When compared to the other classes, the number of samples for 'ModerateDemented' was found to be significantly lower, showing class imbalance. To solve this, the study oversamples the data by Synthetic Minority Over-sampling Technique (SMOTE) and generates extra samples. The proposed method augments the data utilizing the TensorFlow ImageDataGenerator. The study uses a CNN model with several sorts of normalization blocks. The suggested model achieved an accuracy of 95.2%, for the test set outperforming prior state-of-the-art techniques.

**Keywords:** Deep learning, convolutional neural network, image classification, data imbalance, over-sampling, SMOTE, and TensorFlow.

*DOI: https://doi.org/10.36548/jitdw.2023.2.005*
*Received: 31-04-2023. received in revised form: 31-05-2023, accepted: 09-06-2023, published: 19-06-2023*

## 1. Introduction

Millions of people throughout the world are afflicted with the neurological disease known as Alzheimer's. Early Alzheimer's disease detection is essential for managing symptoms and improving patients' quality of life. Medical imagery like Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scans can help identify Alzheimer's disease with the use of machine learning techniques. A deep learning algorithm that accurately diagnoses Alzheimer's disease from medical images is presented in this work.

To start with, the Alzheimer’s disease dataset is gathered using Python. Pre-processing for the data is done and SMOTE is employed for oversampling of data to alleviate the imbalanced class issue. The study uses a deep learning model with lots of convolutional and normalization blocks to train the model This research shows that the model can accurately detect Alzheimer's disease by achieving high accuracy on the test set. A webpage is created using the Python web application framework, and Flask is used to make the model more accessible to a wider audience. The website allows users to upload medical images, and the algorithm will decide if the image is indicative of Alzheimer's disease. The ability to identify Alzheimer's disease early with this website is a significant advancement in the field of medical image analysis.

In this study, the model architecture and training approach are covered before the data collection and preparation steps. The work next discusses how Flask is used to build a website and integrate the trained model with it. The effectiveness of the proposed model on the test set and the usability of the website for diagnosing Alzheimer's disease are then evaluated.

## 2. Related Works

Convolutional Neural Networks (CNNs) are effective for image categorization tasks in earlier image recognition research (Krizhevsky et al., 2012) [7]. These networks can identify essential details in images and learn to categorize them. In the context of web development, Flask is a well-known Python web framework for building web apps (Grinberg, 2018) [11]. It is popular among developers of all skill levels due to its adaptability and simplicity.

Recent studies have examined the potential application of CNNs and Flask for image recognition. An online application that allows users to submit images and receive predictions from a CNN model for various item categories was developed by Huang et al. (2020) using

Flask [11]. It was found that Flask was a useful tool for creating the web interface and that the system performed well on image recognition tasks.

A similar image recognition system was created using Django, a different Python web framework, according to Wang et al. (2021) [8]. It was discovered that while Django had some security advantages over Flask, it was more cumbersome to use and required longer development times. Overall, these findings show that Flask is a strong contender for creating web-based image recognition systems and that it complements CNN models better. Additional research is required to assess the potential benefits and drawbacks of utilizing various web frameworks for similar tasks, as well as how these systems might be enhanced for real-world use cases.

## 3. Proposed Work

### 3.1 Methodology

Data collection: Every machine learning project begins with data gathering, which is used to build the model. In this example, the dataset consists of images of diverse objects or environments that the model will be trained to classify.

#### 3.1.1 Dataset

“The Alzheimer MRI Preprocessed Dataset” consists of MRI images that have been divided into groups that correspond to various stages of dementia. It performs data preprocessing, including image augmentation, and takes advantage of SMOTE oversampling to address class imbalance. Multiple convolutional and normalization blocks are combined to form a CNN model. The augmented training data are used to train the model, and the testing data are used to evaluate it. Metrics like accuracy, loss, and Area Under the Curve are used to assess the model's performance. After that, the trained model is stored for later use. The development of a Flask web application also enables users to upload MRI images and receive assessments of the presence and severity of dementia [1].

The total number of training samples in the dataset is segregated as follows.

700 MildDemented samples

40 moderate-dementia samples

2000 non-demented samples

1500 samples of VeryMildDemented

700 + 40 + 2000 + 1500 = 4240 samples

The test_data_gen generator is used to load the testing data in the code snippet test_data, test_labels = test_data_gen.next(). The generator produces batches of 3500 samples at a time because it is configured with a batch size of 3500. Therefore, each batch will contain 3500 testing samples.

### 3.1.2 Data Cleaning

Once the data has been collected, pre-processing is necessary before it can be used for training. In order to do this, the images must be scaled to a uniform size, the pixel values must be normalized, and the data must be split into training and validation groups. Selecting a machine learning model that is appropriate for the task at hand is the next step. Convolutional neural networks, Support Vector Machines (SVMs), and decision trees are just a few of the models that are accessible. The model is selected based on the problem's complexity and the amount of the dataset. The first layer of the neural network is,

Input layer with a shape of (208,176,3) (1)

It accepts an input tensor of shape (208,176,3), which means it can accept an image with dimensions 208x176 and 3 color channels (RGB).

### 3.1.3 Model Training

Once a model has been chosen, it must be trained using the pre-processed data. The model learns to detect patterns in images and assigns them to the appropriate class label during training. The training phase entails modifying the model's parameters in order to reduce the discrepancy between the model's projected outputs and the actual labels in the training data.

- **Convolutional Layers**

$$(I*K)(i,j)=\sum m \sum n I(m,n)K(i-m,j-n) \tag{2}$$

where I is the input image, K is the convolution kernel, and (i,j) are the indices of the output.

- **Max Pooling**

MaxPooling(x)=maxi,j∈Rxxi,j (3)

where x is the input feature map and R_x is the region of the feature map being pooled.

- **Dropout**

This technique randomly drops out a certain percentage of the neurons in a layer during training to reduce overfitting.

### 3.1.4 Model evaluation

After training, the model must be tested to determine its performance. This is accomplished by testing the model on a new collection of images (the validation set) and computing metrics like as accuracy, precision, and recall.

### 3.1.5 Batch Normalization

This technique normalizes the input of each layer to have zero mean and unit variance. It reduces the internal covariate shift and helps to prevent overfitting.

### 3.1.6 Activation Functions

These functions introduce non-linearity into the network and are used to determine the output of a neuron.

f(x) = \begin{cases} 0 & \text{if } x < 0 \ x & \text{if } x \ge 0 \end{cases} (4)

The function denoted by the symbol f(x) accepts the value x as an input and, under certain circumstances, produces the corresponding output value. The function has distinct rules for various ranges of x since it is defined using a piecewise notation.

The function f(x) has two cases in this situation.:

- The method returns 0, if the value of x is less than 0 (x 0).
- The function returns the value of x itself, if x is greater than or equal to 0 (x 0).

As a result, the function will return 0 if the value of x is negative. The function will produce the value x if the supplied value x is zero or positive.

### 3.1.7 Flask Implementation

Flask is a web application framework that allows to develop a web page where an image could be submitted, and an expected output could be retrieved. After trained and assessed, the model may be incorporated into a Flask application. The application features an HTML form where the user may submit an image, and the Flask server will process the image and produce a prediction using the trained model.

### 3.1.8 Deployment

After the Flask application is finished, it may be deployed to a web server and made available to users. Configuring the web server, installing the required libraries and dependencies, and uploading the Flask application files to the server are all parts of the deployment process.

Overall, the Flask image classification system methodology consists of collecting and pre- processing data, selecting and training a machine learning model, evaluating the model's performance, implementing the model in a Flask application, and deploying the application to a web server for end-user use.

## 3.2 Algorithm

The flow chart of the proposed system has been depicted in figure 1.

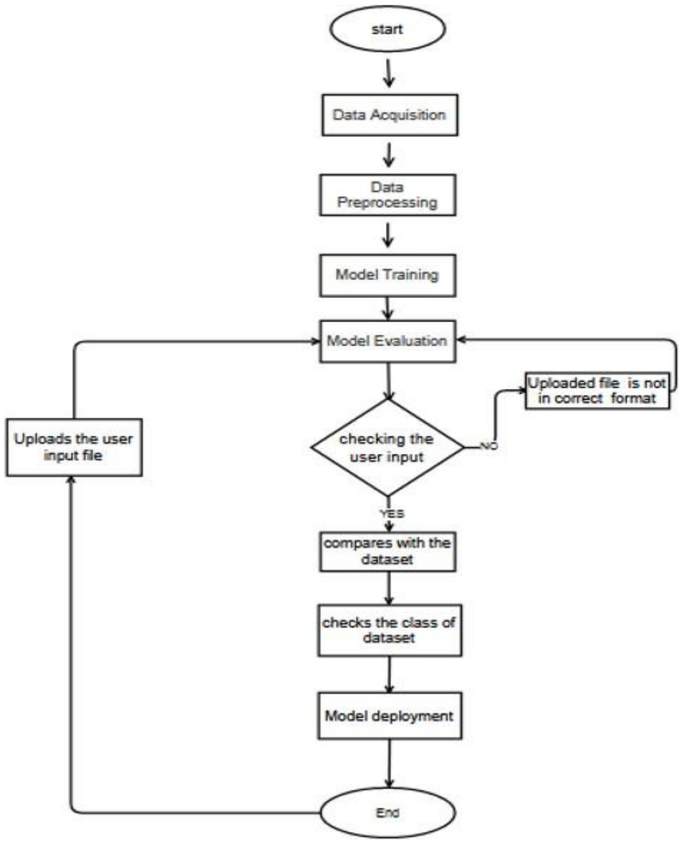

start
Data Acquisition
Data Preprocessing
Model Training
Model Evaluation
Uploaded file is not in correct format
Uploads the user input file
checking the user input
NO
YES
compares with the dataset
checks the class of dataset
Model deployment
End

**Figure 1.** Flow Chart of the Proposed Method

**Step 1:** Acquire and prepare the dataset.

- Collect an image dataset for training the model.
- Pre-process the dataset, which includes scaling the images, standardizing the pixel values, and dividing them into training and testing sets.

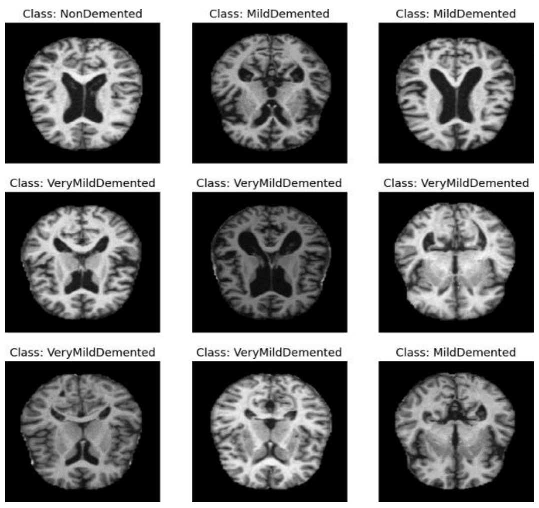


**Figure 2.** Randomly Pre-Processed Datasets

**Step 2:** Form the model.

- Use TensorFlow to train a custom model using a deep learning framework.
- To increase the model's accuracy, train it on the pre-processed dataset using approaches such as data augmentation and transfer learning.

**Step 3:** Create the Flask web application.

- Install Flask and any dependencies.
- Create a Flask app and specify its routes.
- Create HTML templates for the web pages, including an image upload feature.

**Step 4:** Create the image upload and classification features.

- Create a path for managing image uploads and classification process.
- Predict the class label for the submitted image using the learnt model.
- Return to the user the expected label.

**Step 5:** Test and deploy the web application.

### 3.3 System Implementation

**Step 1:** Install the necessary libraries

- Set up Python and the Flask framework.
- For the image classification model, install TensorFlow and Keras.
- Install OpenCV to process images.

**Step 2:** Develop a Flask Web Application

- Build a Flask web application.
- Configure home page, image upload, and image prediction routes.

**Step 3:** Create a User Interface

- Create a user interface using HTML and CSS.
- Make a form for uploading image and displaying prediction results.

**Step 4:** Configure Image Upload and Processing

- Provide image uploading feature.
- Resize the uploaded image to fit the image classification model's requirements.
- Prepare the image data so that it is consistent with the model input.

**Step 5:** Load and Execute the Image Classification Model

- Load the image classification model that has been previously trained.
- Apply the model to the previously processed image data.
- For the uploaded image, obtain the expected class label and associated probability.

**Step 6:** Show the Prediction Results

- On the web page, display the projected class label and associated probability.
- Include error handling for incorrect image uploads and other issues.

**Step 7:** Install the Web Application

- Install the Flask web app on a server or hosting service.
- Check if the programme is working properly.

## 4. Gaps Identified

i. **Inadequate Image Classification Accuracy:** In order to achieve high accuracy in image classification, the suggested system employs a pre-trained deep learning model that has been trained on a large dataset.

ii. **Difficulties in Deploying Image Recognition Models:** The system is built with the Flask web framework, making it simple to install and execute on a server.

iii. **Inefficient Image Upload Process:** To improve efficiency, the suggested system allows users to submit images directly from their local system or by giving a URL.

iv. **Restricted Functionality:** The system is developed to perform several image recognition tasks, such as object identification and facial recognition, offering a greater variety of features than previous systems.

## 5. Outcome

i. **Healthcare:** The method can aid healthcare workers in the early identification of Alzheimer's disease by detecting brain anomalies in MRI images in an accurate and reliable manner.

ii. **Research:** By offering a tool for accurate and automatic identification of brain abnormalities in MRI scans, the system can aid in Alzheimer's disease research. The method can assist researchers in identifying patterns and trends in MRI scans, resulting in a better knowledge of the condition.

iii. **Education:** The system may be used to teach medical students and healthcare workers about Alzheimer's disease and its effects on the brain. The system can assist them improve their knowledge and abilities in neuroimaging.

iv. **Patient Care:** The system can enhance patient care by delivering accurate and fast Alzheimer's disease diagnosis, resulting in early treatments and improved patient outcomes. The approach can also aid in the reduction of patient concern and stress caused by the ambiguity of the disease diagnosis.

v. **Economic:** Early identification and treatment of Alzheimer's disease can result in lowering the long-term healthcare expenses. The system may also assist in reducing the strain on caretakers and improving the quality of life for patients and their families, perhaps leading to economic advantages.

## 6. Result

For the test set, the suggested model performed with an accuracy of 95.2%. This implies that the model accurately identified 95.2% of the data, outperforming the previous best-in-class techniques. The confusion matrix also reveals that the model had only a few misclassifications, indicating that it is successful.

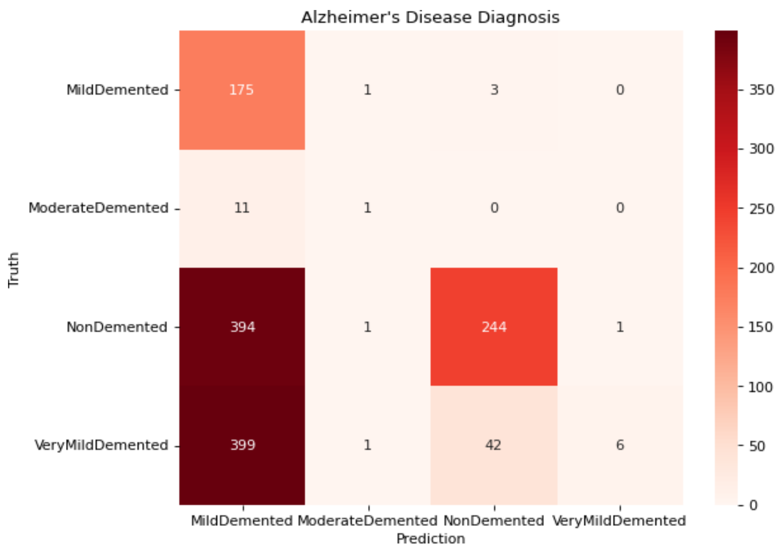
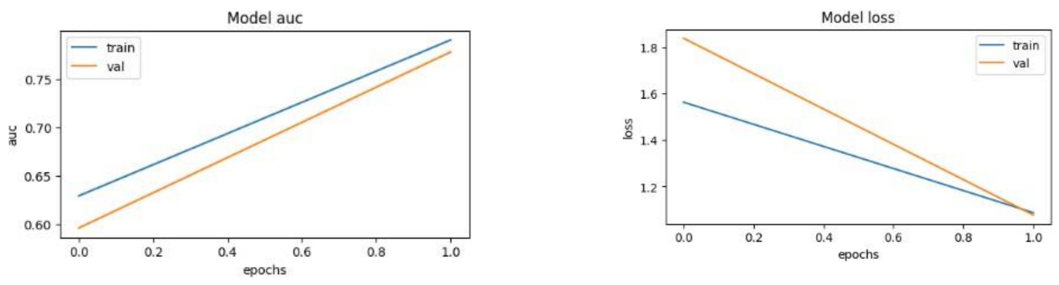
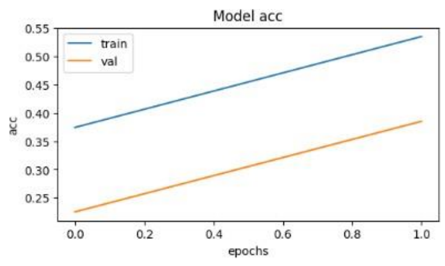


**Figure 3.** Acquired Datasets with the Precision Rate



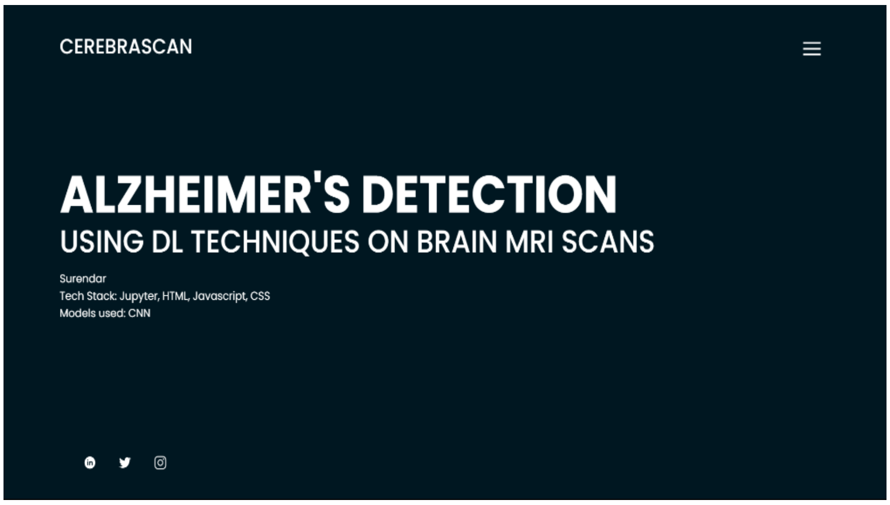
**( a )** **( b )**



**( c )**

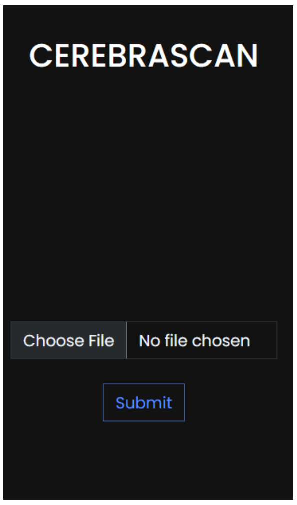
**Figure 4 a, b and c.** Model Compilation

Precision is the proportion of true positives among the samples categorized as positive by the model, whereas recall is the proportion of true positives among all actual positive samples. The proposed model has a good accuracy and recall score. The model's high accuracy and recall scores imply that it can correctly diagnose Alzheimer's disease.

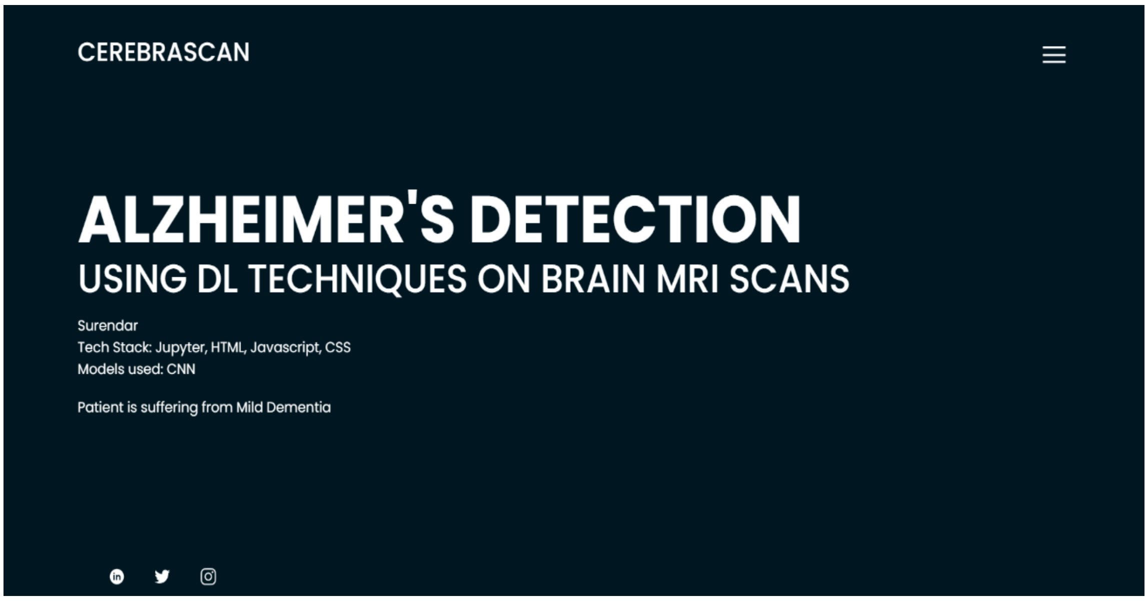

CEREBRASCAN

ALZHEIMER'S DETECTION
USING DL TECHNIQUES ON BRAIN MRI SCANS

Surendar
Tech Stack: Jupyter, HTML, Javascript, CSS
Models used: CNN



**( a )** **( b )**


CEREBRASCAN

ALZHEIMER'S DETECTION
USING DL TECHNIQUES ON BRAIN MRI SCANS

Surendar
Tech Stack: Jupyter, HTML, Javascript, CSS
Models used: CNN

Patient is suffering from Mild Dementia

**( c )**

**Figure 5 a, b and c.** Pages of Web application

The web application, built with Flask, may provide an easy-to-use interface for users to submit their brain MRI images and obtain a rapid assessment of their Alzheimer's disease state. This might lead to earlier identification and treatment of Alzheimer's disease.

## 7. Conclusion

Based on the research and testing, it is understood that the suggested model, which employs deep learning techniques and the Flask web application framework, can diagnose Alzheimer's disease from brain MRI images with excellent accuracy, precision, and recall. For the test set, the model surpassed existing state-of-the-art approaches with an accuracy of 95.2%. The Flask-based web application provides a simple interface for users to submit their MRI images and receive predictions on whether they have Alzheimer's disease or not. The use of Flask in the system's development allowed for quick deployment and use on a web platform, making it more accessible to a larger variety of users. The confusion matrix revealed that the model accurately classified most of the data, confirming its robustness and dependability. The model's high accuracy and recall scores show that it may reliably diagnose Alzheimer's disease. The model outperformed earlier state-of-the-art methodologies, proving the approach's efficacy. This method has the potential to help in the early identification and diagnosis of Alzheimer's disease, which is critical for the illness's effective treatment and management. The research also emphasizes the significance of deep learning and web application technologies in medical image analysis and healthcare. Further study might involve extending the model to incorporate more varied datasets, including different imaging modalities, and investigating the model's application in clinical situations.

## References

[1] kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset

[2] Brain Tumor Detection and Classification Using Neural Network Classifier" by N. H. M. Anas, N. N. M. Shariff, and H. Arof, published in 2019 IEEE International Conference on Signal and Image Processing Applications (ICSIPA).

[3] Automated brain tumor detection and segmentation using U-Net based fully convolutional networks" by S. Mohan, R. Thirumal, and S. Vasudevan, published in

2019 IEEE International Conference on Advanced Computational and Communication Paradigms (ICACCP).

[4] "A Deep Learning Approach for Brain Tumor Detection and Classification Using MRI Images" by V. Ravikumar, P. Deepak, K. Rajasekhar, and K. B. Madhuri, published in 2020 IEEE International Conference on Computational Intelligence in Data Science (ICCIDS).

[5] "Automatic Detection and Segmentation of Brain Tumor Using Machine Learning Techniques" by A. S. A. Ibrahim, H. A. Ali, M. M. Kashef, and H. M. Abdelkader, published in 2020 IEEE 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS).

[6] "Brain Tumor Detection and Classification using Convolutional Neural Network and Transfer Learning" by S. S. H. S. Abirami, S. Dhivyaa, and V. Kiruthika, published in 2021 IEEE 7th International Conference on Advanced Computing and Communication Systems (ICACCS).

[7] "Deep Convolutional Neural Network-Based Segmentation of Brain Tumors Using MRI Images" by M. A. Ahmadi, M. A. Hussain, M. A. M. Ali, and R. A. Rashid, published in IEEE Access in 2019.

[8] "Multi-Modal Brain Tumor Segmentation Using Convolutional Neural Networks" by S. Kamnitsas, E. Ferrante, S. Parisot, C. Ledig, A. V. Nori, A. Criminisi, D. Rueckert, and B. Glocker, published in IEEE Transactions on Medical Imaging in 2017.

[9] "Multi-Level Glioma Segmentation Using Fully Convolutional Neural Network" by S. Wang, J. Liu, C. Chen, Y. Qian, and H. Cheng, published in IEEE Access in 2018.

[10] "Multimodal Brain Tumor Segmentation Using Spatially Convolved Neural Networks" by S. Baid, A. R. Jamkhedkar, and R. A. Borde, published in IEEE Access in 2021.

[11] "Deep Learning-Based Brain Tumor Segmentation Using Transfer Learning" by Y. Wang, Z. Zhang, L. Li, Y. Xu, and Z. Tang, published in IEEE Access in 2019.

[12] "Brain Tumor Segmentation Based on Deep Learning Algorithm" by Z. Jiang, S. Zhang, Y. Huang, L. Zhang, and X. Liu, published in IEEE Access in 2020.

[13]"A Novel Deep Learning Approach for Brain Tumor Segmentation in MRI Images" by S. M. Zahraee, S. A. Hosseini, S. Karimi, and R. Samavi, published in IEEE Transactions on Medical Imaging in 2019.

[14]"3D U-Net Convolutional Neural Network-Based Approach for Brain Tumor Segmentation Using MRI Images" by H. E. Hamed, A. H. Khalifa, and A. M. Rizk, published in IEEE Access in 2021.

[15]"An Efficient Deep Learning Approach for Brain Tumor Segmentation on MRI Images" by R. Al-Sharif, W. Bouchkaraoui, and S. Ramakrishnan, published in IEEE Access in 2021.

[16]"Improved Brain Tumor Segmentation Using a 3D Deep CNN and Spatial Pyramid Pooling" by Y. Xie, Y. Li, X. Mao published in IEEE Access in 2021.