Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 2 (2024)

Classification of Alzheimer's Disease Using Deep Learning based Edge Detection and Fuzzy Neural Network

Nagarathna C R[1], Archana B[2], Anjaneya L.H[3], Mahantesh U[4], Nandini G[5], Babu Kumar S[6]

[1,4,5]Associate Professor, BNM Institute of Technology, Bangalore, Karnataka, India
[2]Associate Professor, VVIT, Mysore, Karnataka, India
[3]Associate Professor, Bapuji Institute of Engineering and Technology, Davangere, Karnataka, India
[6]Assistant Professor, Christ University, Bangalore, Karnataka, India

Abstract:
Purpose: Alzheimer's disease is a neurological ailment that gradually decreases a person's capacity for daily tasks, memory, and reasoning.
Methods: This paper presents the use of a deep learning-based edge detection method called holistically-nested edge detection for Alzheimer's disease categorization. The processed images are fed into a fuzzy neural network's convolutional architecture in order to extract features from the previously processed images. To achieve better categorization, a fuzzy interface system is then used. For several picture kinds, this system explores rich feature representation.
Results: The average accuracy using the MRI dataset is determined to be 95.75% when tested against the dataset including several categories of images related to Alzheimer's disease.
Conclusion: The simulation result exemplifies that proposed method can be capable to determine ideal global solutions efficiently with exactly than existing methods.
***Keywords**:* Alzheimer's, Fuzzy neural network, Edge detection.

1. Introduction

Soft computing, in contrast to conventional computing, addresses approximation techniques and offers solutions for challenging real-world issues. Soft computing, as opposed to hard computing, allows for approximations, vulnerability, imprecision, and imperfect truth (Senthilkumaran and Rajesh 2009). The human mentality is, in fact, a good example of soft computing. Techniques including fuzzy logic, artificial neural networks (ANN), expert frameworks, genetic algorithms (GA), and artificial intelligence (AI) are used in soft computing. The application of imprecise approximations to provide ambiguous yet practical solutions for intricate computer problems is known as soft computing (Shajin et al. 2022; Saridakis and Dentsoras 2022). The data-driven productivity of neural networks and the flexibility of fuzzy logic are combined in another field of study known as "soft computing" to speak to subjective data. This makes it possible to apply genetic algorithms to achieve effective coarse-granule global searching and to create precisely tailored alterations through local search (Rajesh et al. 2022; Shajin et al. 2022). In the end, fusion algorithms have improved to the point where they outperform all of their core soft computing components and provide superior real-world problem-solving capabilities (Rajesh et al. 2022). Yet, there are a few significant issues that need to be addressed in order to ensure that the computerized process of determining the health of a brain segment is carried out effectively. A number of techniques are offered for the detection and classification of AD

This work primarily focuses on developing a conjoined system of feature extraction, a data categorization system, and lastly a holistically-nested edge detection (HED) system to diagnose Alzheimer's disease. Segmenting the images comes first in the suggested process, then feature extraction using convolutional networks, and classification performance using a fuzzy logic system. Identifying AD and classifying it using neuro fuzzy logic (NFL), a soft computing technique, is the main objective of this work.

The remaining portions of this manuscript are organized as follows: Section 2 discusses the existing work; Section 3 illustrates the recommended technique; Section 4 shows the outcomes; Section 5 shows the discussion; and Section 6 shows the conclusion.

2. Literature Survey

An extensively used technique for AD diagnosis is MRI. According to earlier research, AD patients had less hippocampal and grey matter than healthy individuals. However, the cross sections of sliced photographs and even little variations in image registration can have a significant effect on volume, making the assessment of regional brain volume challenging. When comparing AD patients to non-AD persons, ocular examinations may show that the cerebral ventricle and brain shape alter, becoming more rounded. If the MRI diagnostic's sensitivity for AD detection is higher than volume changes' and if shape modifications can be accurately assessed, then this approach may prove helpful in the early diagnosis of the illness. All of the AD and Health Control (HC) data must match in order to compare specific images and their locations (Fuse, H. et al. 2018).

Using a student's t-test-based feature selection strategy, researcher Acharya U.R extracts features from the MRI dataset that was received from the University of Malaya Medical Centre. The KNN fared better in the data classification process than other feature selection strategies, with a 94.55% classification accuracy. Shearlet Transform (ST) feature extraction improved performance. An Open Access Series of Imaging Studies (OASIS) MRI dataset was used to autonomously diagnose AD disease using a Deep Learning (DL) network ensemble model. The system's accuracy was 93%. The author used the stochastic gradient descent optimization technique to lower network loss and improve system performance (Islam, J 2018).

Using an adaptive histogram, the researcher preprocessed brain MRI data from the Alzheimer's Disease Neuro Imaging (ADNI) dataset. The essential characteristics are extracted using Student's t-test and Adaptive synthetic sampling (ADASYN) after the dataset has been pre-processed for balance. Then, an accuracy of 93.9% was obtained for each Random Forest (RF) and SVM-Poly 1 classifier employed for classification (Islam, J 2018). To determine whether it would be possible to distinguish between AD, MCI, and NC using the MRI ADNI dataset, the CNN model was employed. The Talairach transform, motion correction, non-uniform intensity normalization, intensity normalization, and skull-stripping procedures were employed by the author to convert 3D MRI data into a 2D slice. The characteristics are obtained after 2D MRI slices are obtained so that the CNN model may be used to classify the data. Koh, J. et al. (2020) reported that the model's accuracy for AD vs. NC classification was 94.97%, and for AD vs. MCI classification, it was 91.98%.

In order to detect AD using MRI, Braulio Solano-Rojas created a three-dimensional DENSENET neural network. To distinguish between five different disease stages—NC, Significant Memory Concern, Early MCI (EMCI), MCI, and Late MCI (LMCI)—the author developed a deep neural network classifier. Metrics generated by the classifier included 0.86 mean accuracy, 0.86 mean specificity, 0.91 micro-average area under the receiver operating characteristic curve, and 16 micro-average mean sensitivity (Islam, J. et al. 2018).

3. Methods

Figure 1 depicts the suggested HED and Fuzzy Neural Network (FNN) model. Firstly, pre-process the brain MRI dataset using techniques such as edge identification, mask creation, brain part cropping, and normalization to boost scheme effectiveness. Separate the dataset even more into train and test subsets. Make uses the train data set so that the fuzzy neural network may learn the pre-processed photos. Utilizing the test dataset, assess the FNN model in the end.

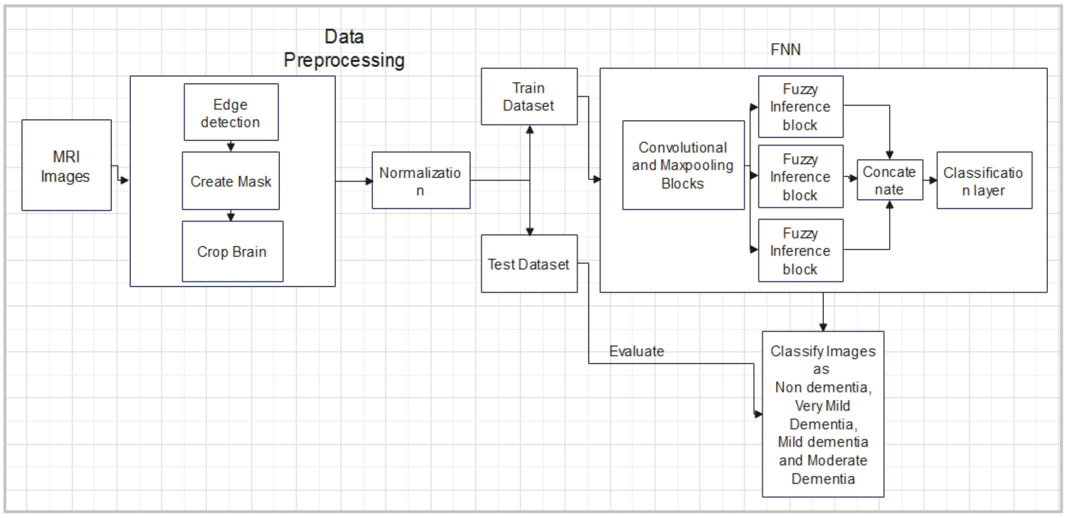


Figure 1: Proposed HED and FNN

Algorithm 1 reveals the intricacies of the suggested task. The images are subjected to DL-based edge detection, or HED, before being mask-created. For the MRI scans with numerous orientations, additional image augmentation and normalization are conducted in order to increase the number of images in the collection. Next, using the masked images, the largest contours are located. Following that, a bounding box is made, the image within of which is cropped, and it is saved for later classification.

Algorithm 1: Classification of AD using DNN and holistically-nested edge detection

Input: Brain Images

Output: Classified images

Step 1: Import images.

Step2: Pre-processing images

• Edge detection using DL.

• Use the edge detected image to create a mask image.

• Use the masked image to locate the largest contours.

Make a bounding box.

• Save the cropped image after cropping it within the enclosing box.

Step 3: FNN for Feature Extraction

Step 4: Use of a fuzzy interface mechanism for classification.

Step 5: Use the acquired feature vector to train the FNN model.

Step 6: Approve test pictures and note the outcome.

Step 7: Identify and categorize the pictures.

This study attempts to demonstrate the use and implementation of DL grounded edge detection, or HED. This study is being used to categorize the Kaggle MRI dataset into four stages. It combines feature extraction, classification, and HED techniques.

3.1 Image Processing

Dataset: The MRI images from publicly accessible datasets are explored with in this planned effort. The MRI images produced by researcher Sarvesh Dubey from (Dubey 2020) Kaggle. The four phases of AD represented in the dataset are Moderate Demented (MOD-3), Very Mild Demented (VMD-1), Mild Demented (MID-2) and Non-Demented (ND-0). Figure 1 displays the total number of photos for each phase as well as sample photographs for each dataset class. The ND VMID, MID, and MOD samples are displayed in the 0, 1, 2, and 3 accordingly. Table 1 displays the number of Kaggle Datasets.

Table 1: Kaggle Dataset number

Classes	Kaggle_Dataset_Count
ND	3200
VMD	2240
MID	896
MOD	64

Edge Detection: One of the first practical uses of CV was edge identification, which facilitates the identification of the borders of objects in the pictures. The Canny edge identifier is most likely the mechanism utilized when utilizing OpenCV to detect an object's edges. It should be noted that there are a few drawbacks to employing Canny Edge for edge detection, such as the need for hysteresis thresholding and additional pre-processing steps like concealing and switching to greyscale in order to obtain good edge maps. HED is used to manage the aforementioned problems.

Open CV has integrated a form of DL-based edge detection technology into its ostentatious new DNN module. Holistically nested edge detection, or HED for short, is the strategy employed in this. It is an edge detection system that operates from start to finish and is based on learning. 3.4.2 or a later version of Open Cv is needed to use this new technique. The system/network receives an RGB image as input. The edge map, the network's output, performs better than the original by capturing and maintaining an object's borders within an image. HED makes use of intermediate layer yields. The term "side output" describes the yield from earlier layers that are combined to get the final prediction. The feature maps generated at each layer vary in size, therefore it is possible for it to view the provided image at several different scales.

The HED strategy outperforms other deep learning-based strategies not just in terms of accuracy but also in terms of speed. "Why Open CV is chosen for coordinating into new DNN component" stems from this rationale. It is a framework for end-to-end edge identification. As a result, HED can become proficient in the kind of rich, varied leveled features that are essential to advancing human ability to assess uncertainty in natural picture boundary detection of an item and edge. Since HED plans to prepare and anticipate edges in a image to image manner, even though it does not explicitly model the structured output, the name holistic is used. Because it highlights the side outputs that are generated that correspond to gradually and inherited refined edge maps, the phrase "nested" is employed. The predictions established along the route are fundamental and common for each edge map that is generated. More compactness is added to the progressive edge maps. The coordinated learning of several leveled aspects distinguishes current multi-scaled techniques (Yuille and Poggio 1986; Ren 2008; Whitkin 1984).

The anticipated HED addresses two important issues:

A. Using FCNN-powered training and prediction to do image-by-image classification (the framework takes an image as input and produces edge maps as output in an easy-to-understand manner).

B. Nestled multiple scale feature learning is a notable attribute that is driven by deep learning networks (Long et al., 2015), which carry out deep layer oversight to help with earlier yields. The advantages that these foundational approaches exhibit in HED are found to be accurate and computationally efficient.

Let's use S to represent the input training data set for the training phase, S= $\{(X_n, Y_n), n = 1, \dots, N\}$, where sample $X_n = \{ x_j^{(n)}, j = 1, \dots, |X_n|\}$ and $Y_n = \left\{y_j^{(n)}, j = 1, \dots |X_n|\right\}, y_j^{(n)} \epsilon \{0,1\}$ depict both the original and equivalent ground truth binary edge maps for a image X$_n$. Because each picture or image is viewed holistically and independently, the subscript "n" is removed to maintain the simplicity of the notation. The aim was to create a model of a system that processes inputs and generates edge maps that are in proximity to ground realities. For simplicity, let us assume that W is the collection of all network layer standard parameters. Let us assume that there are m side-yield layers in the system. Additionally, each side-yield layer was linked to the classifier, where the corresponding weights were intended, as$w = (w^{(1)}, \dots, w^{(M)})$. The goals are gauged as follows:

$$L_{side}(W, w) = \sum_{m=1}^{M} \alpha_m l_{side}^{(m)}\left(W, w^{(m)}\right) \tag{1}$$

when, The loss function for side outputs at the picture level is indicated by ℓside. Every pixel in a training image and edge map is subjected to a loss function processing during the HED training process. Delivery of edge and non-edge pixels is strongly biased for a typical characteristic picture. Long et al. (2015) implemented a cost-effective loss function and provided additional compromised limitations for biased sampling. Instead, a simpler mechanism that automatically modifies the loss between positive and negative classes is applied. Additionally, each pixel base has a class-adjusting weight β shown. Next, use this weight that adjusts for class as an easy way to even out this imbalance between edge and non-edge. The accompanying "class-adjusted cross-entropy loss function" is specifically described as follows:

$$\begin{aligned} L_{side}^{(m)}\left(W, w^{(m)}\right) = -\beta \sum_{j \epsilon Y_+} logPr\left(y_j = 1 \middle| X; W, w^{(m)}\right) - (1 \\ - \beta) \sum_{j \epsilon Y_-} logPr(Y_j = 0|X; W, w^{(m)}) \end{aligned} \tag{2}$$

where $\beta = \frac{|Y_-|}{|Y|}$ and $1 - \beta = \frac{|Y_+|}{|Y|}$. $|Y_-|$ and $|Y_+|$ denotes "ground truth label sets with edges and non-edges," respectively.. $\Pr\left(y_j = 1 \middle| X; W, w^{(m)}\right) = \sigma\left(a_j^{(m)}\right) \epsilon [0,1]$is calculated by means of "sigmoid function σ(.) over activation value present in pixel j". At every side-yield layer, acquire edge maps prediction$\hat{Y}_{side}^{(m)} = \sigma\left(\hat{A}_{side}^{(m)}\right)$, where $\hat{A}_{side}^{(m)} \equiv \left\{a_j^{(m)}, j = 1, \dots, |Y|\right\}$ are activation of side-yield of m layer.

Incorporating a weighted combination layer into the system and simultaneously examining the combination weight during training will enable the straightforward application of side-yield expectations. The fusion layer's loss function, L_fuse, develops as follows:

$$L_{fuse}(W, w, h) = Dist(Y, \hat{Y}_{fuse}) \tag{3}$$

where $\hat{Y}_{fuse} \equiv \sigma \sum_{m=1}^{M} h_m \hat{A}_{side}^{(m)}$ where the cross entropy loss is indicated by Dist (·, ·) and h = (h1,..., hM) is the fusion weight. After everything is put together, use "standard stochastic gradient descent" to limit the accompanying objective function:

$$(W, w, h)^* = argmin\ (L_{side}(W, w) + L_{fuse}(W, w, h)) \tag{4}$$

In order to obtain edge map prediction from weight fusion and side yield layer, first receive image X as an input during testing. This is indicated by:

$$\left(\hat{Y}_{fuse}, \hat{Y}_{side}^{(1)}, \dots \dots \dots, \hat{Y}_{side}^{(m)}\right) = CNN(X, (W, w, h)^*) \tag{5}$$

Here, CNN (·) denotes an edge map that was obtained using the system. By taking an additional average of these generated edge maps, the final united yield may be obtained.

$$\hat{Y}_{HED} = Average(\hat{Y}_{fuse}, \hat{Y}_{side}^{(1)}, \ldots\ldots, \hat{Y}_{side}^{(m)}) \tag{6}$$

As a result, feedback must be spread via the middle layers since the multi-scale reactions generated at hidden layers at (Hwang and Liu 2015) (Bertasius et al. 2015) are low significant due to the lack of deep supervision, which is recalled for the proposed approach. Moreover, their patch-to-patch or patch-to-pixel strategies seriously impair the efficacy of both training and prediction. This work intends to demonstrate an end-to-end edge detection framework and a system that uses "holistically-nested" architecture to provide additional deep supervision on top of the convolutional model, triggered by an FCNN system. Figure 2 presents the images obtained by applying the HED approach.This image is the input image that the NN system will use to make an inference.

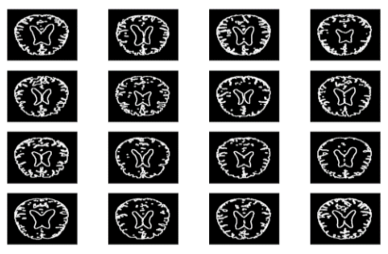


Figure 2: Output of HED

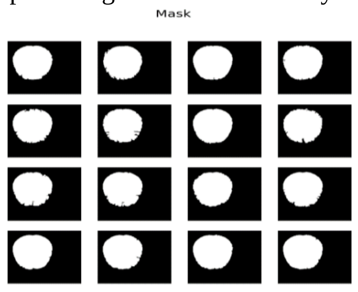


Figure 3: Masked image

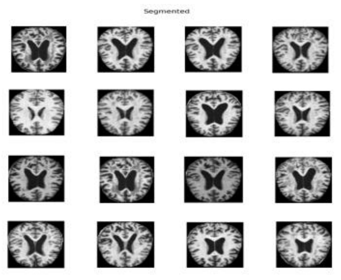


Figure 4: Cropped image

The scalefactor is used to scale the provided images. Multiply the constant number by 255 and divide the result to achieve this. Each pixel's value so falls between 0 and 1, or 0/255-255/255.In the event that scaling is not there, set the default value to 1.0. The scaling factor for the model used in this investigation is 1.0.

size: Given the result, the corresponding spatial size is given. The size will typically be similar to the input size needed for the next generation neural network to yield. Both height and width are utilized by the model.

swapRB: Boolean indicating the need to alter the primary and final channels of an RGB image. Assume that all of the photographs on your open CV were originally in BGR format. Set the flag to 'True,' which is the default, and convert this sequence to RGB if the other direction is required. The model assumes that this parameter is "False."

mean: To account for fluctuations in the intensity condition and standardization, the average pixel value over the training dataset must be computed. Each image at the time of training must then be further subtracted or withdrawn. Mean subtraction must be used during inference if it is applied during training. The determined mean is generally composed of R, G, and B channel tuples. Using swapRB=False will result in this sequence being (B, G, R). In the current study, the factor sets for R, G, and B are 104.00698793, 116.66876762, and 122.67891434, in that order.

crop: It's a flag that allows you to center-crop your photos using Boolean values. Setting this parameter to "True" will crop the input image in the center so that the smaller dimensions match the corresponding size and the other measurement matches or exceeds the larger measurement. When set to "False," on the other hand, it preserves the feature proportion and just resizes to the specified dimensions. The value of this option is set to "False" in the current study.

Use the edge-detected image to create a mask: Masking is nothing more than setting a portion of a picture's pixel values to zero or another value for the background. A tiny area of the image is defined and used during masking in order to modify a larger image. The method that lies behind several types of image processing, including edge detection, is called masking. The masking approach performed to the dataset is displayed in the image below. It provides the region of interest while hiding the background. The rotated image with mask is shown in Figure 3. To accomplish data augmentation, each image in the dataset is rotated many times.

Tracing the perimeter of the box and resizing the picture inside: The bounding box provides the region of interest, and the photos need to be cropped before being passed along to provide the model with the most accurate region of interest. Figure 4 displays the cropped photos according to the bounding box boundaries.

3.2 Feature extraction Model

The goal is to create a DNN system that can generate many leveled features with proficiency and that can capture different phases at different speeds in order to capture typical edge-map sizes.

For feature extraction and classification in the planned work, a fuzzy neural network model is taken into consideration. The fuzzy neural network with five convolutional layers and maxpooling layers for feature extraction is taken into consideration here. It is thought to be able to achieve remarkable depth in performance neural network design by employing five convolutional layers. Nonetheless, numerous contemporary researchers have demonstrated via their research that fine-tuning the pre-trained DNN model on the classification of common images is beneficial for edge detection at the lower level. The planned research embraces this, but with a few changes that are outlined below:

(a) FCN layers are trimmed.

(b) The fuzzy classifier is coupled to the output layer. The related responsive field's dimension is identical to that of the corresponding side-yield layer.

The input volume is convolved by using the weights volume. The input layer is shrunk or expanded based on the padding and out striding. Lessen the width and height of the space. On the other hand, deepen the convolution process. In order to obtain a more complex target function, a non-linear activation function is used to each layer. The complex function need to exhibit non-linear variation in response to the given input image. The feature extraction method's layer summary is displayed in the table. The model consists of five convolutional layers with 20, 40, 40, 40, and 3 filters, respectively, with the same padding on the 3X3 and 2X2 strides. Relu activates each neuron in the model. Following the convolutional layers are maxpooling layers with the same padding and 1x1 stride. Table 2 provides the convolutional neural network's full layer summary for the FNN. The table shows each layer's input and output shapes. The network receives input in the format 224X224x3. The final image output shape is 4X4X3.

Table 2: Layer summary of Feature Extraction Model

Layer's description	**Filter**	**Stride**	**Padding**	**Activation**	**Input shape**	**Output Shape**
Convolutional Layer 1	20 Filters, 6X6	3x3	Same	Relu	224x224x3	73x73x3
Maxpooling 1	40 Filters, 3X3	1x1	Same	Relu	73x73x20	73x73x20
Convolutional Layer 2	40 Filters, 6X6	2X2	Same	Relu	73x73x20	34x34x40
Maxpooling2	6X6	1x1	Same	Relu	34x34x40	34x34x40
Convolutional Layer 3	40 Filters, 3X3	2X2	Same	Relu	34x34x40	16x16x40
Maxpooling 3	2X2	1X1	Same	Relu	16x16x40	16x16x40
Convolutional Layer 4	40 Filters, 3X3	2X2	Same	Relu	16x16x40	7X7X40
Maxpooling 4	2X2	1X1	Same	Relu	7X7X40	7X7X40
Convolutional Layer 5	3 Filters, 4X4	2X2	Same	Relu	7X7X40	4X4X3
Dropout	-	-	-	-	4X4X3	4X4X3

3.3 Classification model

The method for classifying the combined qualities that were developed is described in this section using a fuzzy neural network. A specific type of fuzzy scheme called the neural fuzzy scheme uses fuzzy and fuzzy rules to determine learning parameters through the use of a NN-based learning technique. The purpose of processing the input is to obtain feature-wise data. To fuzzify the appropriate feature values, utilize the Fuzzy Membership Function. According to Bobyr et al. (2022), the procedure consists of three steps: fuzzification, NN classifier construction, and defuzzification.

Through data fuzzification, a membership function matrix including the degrees of each class and membership function is created. The membership function $\mathrm{mf}_{i,j}(m_i)$ determines the degree of MF of present pattern i and class j where i^{th} patterns signifies $m_i = x_{i1}, x_{i2}, x_{i3} \dots, x_{ik}$. The input vector y indicates $m = [m_1, m_2, \dots m_N]^T$where T denotes transpose operation. To control MF action during fuzzification, fuzzy logic uses the membership function. In this study, a fuzzy bell-shaped membership that is based on a number of factors is used. The following represents the membership function with fuzzy boundaries:

$$\mathrm{mf}(y; p, q, r) = \frac{1}{1 + \left|\frac{m - r}{p}\right|^{2q}} \tag{7}$$

The control parameters are critical; MF's center is determined by r, its breadth by p, and its slope at crossing sites is controlled by q. Equation (8) contains it.

$$\mathrm{MF}(y) = \begin{bmatrix} \mathrm{mf}_{1,1}(m_1) & \mathrm{mf}_{1,2}(m_1) & \mathrm{mf}_{1,3}(m_1) & \dots & \mathrm{mf}_{1,M}(m_1) \\ \mathrm{mf}_{2,1}(m_1) & \mathrm{mf}_{2,2}(m_1) & \mathrm{mf}_{2,3}(m_1) & \dots & \mathrm{mf}_{2,M}(m_1) \\ \mathrm{mf}_{3,1}(m_1) & \mathrm{mf}_{3,2}(m_1) & \mathrm{mf}_{3,3}(m_1) & \dots & \mathrm{mf}_{3,M}(m_1) \\ \dots & \dots & \dots & \dots & \dots \\ \mathrm{mf}_{N,1}(m_1) & \mathrm{mf}_{N,2}(m_1) & \mathrm{mf}_{N,2}(y_N) & \dots & \mathrm{mf}_{N,M}(y_N) \end{bmatrix} \tag{8}$$

The construction of the NN classifier method is explained in the following phase. First, an MN feature matrix is created from the feature matrix. The neural network (NN), which consists of an input layer, an output layer, and a hidden layer, then uses the vector as its input. The hidden layers and input layers of the system are interconnected, and the output layer units are connected to the hidden layers. Each layer's nodes are connected by weights and biases that are randomly selected before the training process starts. The estimated yield is handled as follows once the inputs are linearly integrated and supplied as I/O to NN:

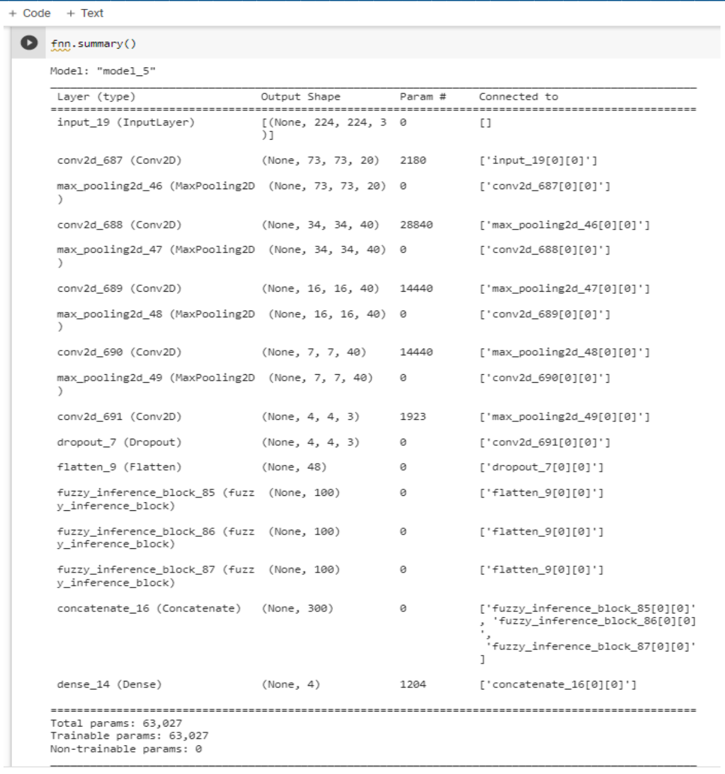
$$\mathrm{Output}_j = \frac{1}{1 + e^{-\mathrm{Net}_j}} \tag{9}$$

The max operator-based hard categorization defuzzification procedure is described. Sort j using the input pattern. It is provided at equation (10).

$$\mathrm{mf}_i(y) \geq \mathrm{mf}_i(y) \forall_i \in (1,2, \dots M) \text{ and } i \neq j \tag{10}$$

There are many advantages to neuro-fuzzy classifiers, such as their capacity to self-sort, self-learn, and self-tune and their lack of requirement for prior knowledge about the relationships between the data. It can calculate more quickly by using fuzzy membership, and it can help with data conflicts and other issues.

In this proposed work designed the fuzzy inference system with three fuzzy inference blocks and one hundred neurons in each inference block. The Equation 9's fuzzy rule function activates the fuzzy neurons. Three layers' worth of output are concatenated and de-fuzzified. Figure 5 displays the snapshot of the FNN model summary.



```
+ Code   + Text

fnn.summary()

Model: "model_5"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_19 (InputLayer)          [(None, 224, 224, 3  0           []
                                )]

 conv2d_687 (Conv2D)            (None, 73, 73, 20)   2180        ['input_19[0][0]']

 max_pooling2d_46 (MaxPooling2D  (None, 73, 73, 20)  0           ['conv2d_687[0][0]']
 )

 conv2d_688 (Conv2D)            (None, 34, 34, 40)   28840       ['max_pooling2d_46[0][0]']

 max_pooling2d_47 (MaxPooling2D  (None, 34, 34, 40)  0           ['conv2d_688[0][0]']
 )

 conv2d_689 (Conv2D)            (None, 16, 16, 40)   14440       ['max_pooling2d_47[0][0]']

 max_pooling2d_48 (MaxPooling2D  (None, 16, 16, 40)  0           ['conv2d_689[0][0]']
 )

 conv2d_690 (Conv2D)            (None, 7, 7, 40)     14440       ['max_pooling2d_48[0][0]']

 max_pooling2d_49 (MaxPooling2D  (None, 7, 7, 40)    0           ['conv2d_690[0][0]']
 )

 conv2d_691 (Conv2D)            (None, 4, 4, 3)      1923        ['max_pooling2d_49[0][0]']

 dropout_7 (Dropout)            (None, 4, 4, 3)      0           ['conv2d_691[0][0]']

 flatten_9 (Flatten)            (None, 48)           0           ['dropout_7[0][0]']

 fuzzy_inference_block_85 (fuzz  (None, 100)         0           ['flatten_9[0][0]']
 y_inference_block)

 fuzzy_inference_block_86 (fuzz  (None, 100)         0           ['flatten_9[0][0]']
 y_inference_block)

 fuzzy_inference_block_87 (fuzz  (None, 100)         0           ['flatten_9[0][0]']
 y_inference_block)

 concatenate_16 (Concatenate)   (None, 300)          0           ['fuzzy_inference_block_85[0][0]'
                                                                 , 'fuzzy_inference_block_86[0][0]
                                                                 ',
                                                                  'fuzzy_inference_block_87[0][0]'
                                                                 ]

 dense_14 (Dense)               (None, 4)            1204        ['concatenate_16[0][0]']

==================================================================================================
Total params: 63,027
Trainable params: 63,027
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 5: Screenshot generated for Proposed Model Summary

The fuzzy neural network model fulfills the need to identify Alzheimer's phases. The four-type Kaggle MRI data set was taken into account.

4 Results

The four distinct classifications from the Kaggle dataset were employed here. Very mild dementia, mild dementia, moderate dementia, and no dementia. This work used a specially configured FNN to classify the brain images. Figure 6 depicts the form of the FNN technique. The method's shape represents the input and output shapes of projected hidden layers. FNN model.

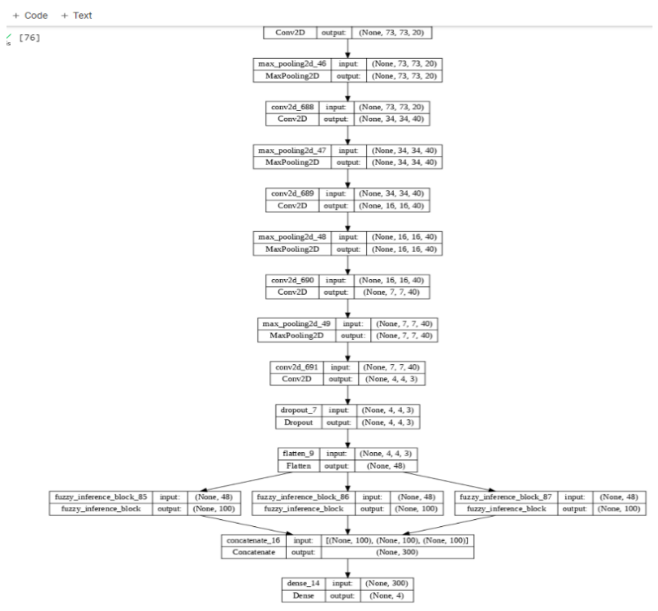

+ Code + Text
[76]
Conv2D | output: | (None, 73, 73, 20)
max_pooling2d_46 | input: | (None, 73, 73, 20)
MaxPooling2D | output: | (None, 73, 73, 20)
conv2d_688 | input: | (None, 73, 73, 20)
Conv2D | output: | (None, 34, 34, 40)
max_pooling2d_47 | input: | (None, 34, 34, 40)
MaxPooling2D | output: | (None, 34, 34, 40)
conv2d_689 | input: | (None, 34, 34, 40)
Conv2D | output: | (None, 16, 16, 40)
max_pooling2d_48 | input: | (None, 16, 16, 40)
MaxPooling2D | output: | (None, 16, 16, 40)
conv2d_690 | input: | (None, 16, 16, 40)
Conv2D | output: | (None, 7, 7, 40)
max_pooling2d_49 | input: | (None, 7, 7, 40)
MaxPooling2D | output: | (None, 7, 7, 40)
conv2d_691 | input: | (None, 7, 7, 40)
Conv2D | output: | (None, 4, 4, 3)
dropout_7 | input: | (None, 4, 4, 3)
Dropout | output: | (None, 4, 4, 3)
flatten_9 | input: | (None, 4, 4, 3)
Flatten | output: | (None, 48)
fuzzy_inference_block_85 | input: | (None, 48)
fuzzy_inference_block | output: | (None, 100)
fuzzy_inference_block_86 | input: | (None, 48)
fuzzy_inference_block | output: | (None, 100)
fuzzy_inference_block_87 | input: | (None, 48)
fuzzy_inference_block | output: | (None, 100)
concatenate_16 | input: | [(None, 100), (None, 100), (None, 100)]
Concatenate | output: | (None, 300)
dense_14 | input: | (None, 300)
Dense | output: | (None, 4)

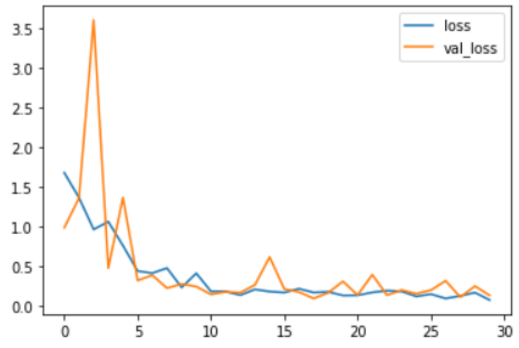
Figure 6: Shape of proposed FNN Model

Table 3 provides values for the Confusion matrix of the examined data. Classifying the stages of the provided brain image into the appropriate illness class is the main objective of the methodology's outcomes, which are now being presented. For test photos of the projected work, the result correctness is shown in Table 3. Using a database of healthy and pathological brain pictures, the suggested approach utilizing FNN architecture is validated. Table 95.75%, 91.42%, 90.98%, 91.10%, 25%, and 74.25% represent average performance for accuracy, precision, recall, F1 score, FPR, and specificity obtained using the predicted technique.

Table 3: Confusion Matrix

Non Demented	**Very Mild Demented**	**Mild Demented**	**Moderate demented**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**FPR**	**Specificity**
284	0	38	22	94.42%	93.42%	82.55%	87.65%	22%	77.8%
0	356	0	0	100%	100%	100%	100%	25%	75%
6	0	345	29	93.58%	86.03%	90.55%	88.23%	30%	70%
14	0	18	321	94.21%	86.29%	90.93%	88.55%	26%	74%
Average				95.75%	91.42%	90.98%	91.10%	25%	74.2%

Figure 7 displays a snapshot of the train dataset and validation dataset's loss and accuracy history graphs. The system is designed to operate with a batch size of 115 over 30 epochs. For the train dataset, the accuracy rate of the system is 97.37%, while for the test dataset, it is 87.75%.



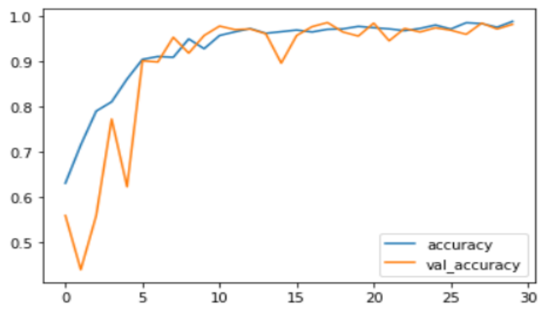
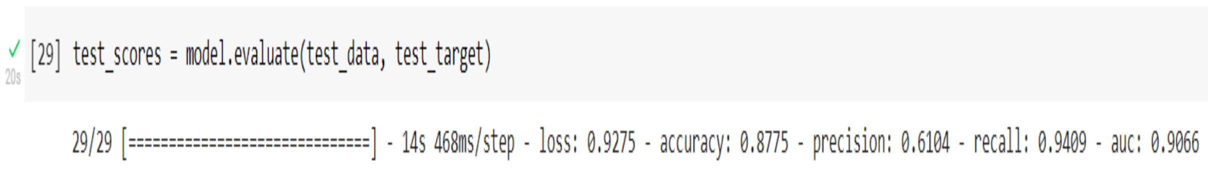


Figure 7: Screenshot of the loss and accuracy history graphs produced by the edge detection model for classification

The historical graphs demonstrate how, with each iteration, the model trains more with the input images, which contributes to a decrease in loss and an increase in accuracy. Figure 8 displays a snapshot of the output produced by the suggested edge detection model, which yields an accuracy of 87.75% for the test dataset.



```
✓ [29] test_scores = model.evaluate(test_data, test_target)
20s

29/29 [==============================] - 14s 468ms/step - loss: 0.9275 - accuracy: 0.8775 - precision: 0.6104 - recall: 0.9409 - auc: 0.9066
```

Figure 8: Screenshot of model performance generated for test dataset

5 Discussion

The statistical performance of the proposed model on a dataset comprising different types of Alzheimer's disease and dividing them into diseased and healthy subjects is presented in the Table 4 displays 128 incorrectly classified images and 1306 successfully classified images based on the statistical performance of the job. Following analysis, the following results were reached: According to this, the model's average performance was 91.25% for precision—the percentage of positive images that are correctly classified—91.25% for sensitivity—the percentage of images that are detected but the remaining 91.25% are not—74.25% for specificity—the percentage of actual negatives that are predicted to be negative—25.75% for false positive rate—the percentage of negative classes that are incorrectly classified—98% for f1-score, and 95.75% for accuracy. In Table 5, you will find the benchmark table.

Table 4: Statistical performance of the work on test images of various AD categories

Classes	TP count	FP count	TN count	FN count	Recall	FPR	Precision
Non-Demented	284	20	1069	60	0.8255	0.022	0.9342
Very Mild Demented	356	0	1077	0	1	0.025	1
Mild Demented	345	56	997	35	0.9055	0.030	0.8603
Moderate demented	321	51	1029	32	0.9093	0.026	0.8296

Table 5: Benchmark Table

Methods	Performance Analysis		
	Accuracy	ROC	Recall
Alorf and Khan (2022)	80.65%	-	-
Savaş (2022)	92.98%	-	
Buvaneswari and Gayathri (2021)	95%	-	89%
Tian et al. (2021)	82.44%	-	
Murugan et al. (2021)	95.23%	87%	-
Diogo et al. (2022)	90.6%	-	-
AD-ED-FNN (Proposed)	95.75%	87.98%	90.98%

The methods that were developed, especially FNN, yielded positive results. This theoretical model has the potential to progress biological research and help with the early detection and treatment of Alzheimer's disease. In the future, we could combine recently developed deep learning models with previously trained deep architectures to obtain more precise results for Alzheimer's diagnosis. Similarly, we will apply deep learning algorithms to the diagnosis of cancer and a few other human disorders.

6. Conclusion

Applications utilizing neurofuzzy techniques have been created to categorize brain images. The work that is anticipated has proved edge recognition based on FNN, also called HED. The main sources of inspiration for the models are deeply directed nets and the head of FCN systems. In FNN architecture, models and structures are used. When evaluated against a dataset comprising multiple categories of images linked to Alzheimer's illness, the average accuracy using the FNN model was found to be 95.75%. The test results show that the selected strategy was a noteworthy technique that could improve the accuracy of Alzheimer's disease diagnosis overall. Apart from the feature extraction procedure, the image is pre-processed, which includes tasks like edge detection, mask creation, cropping masked images, binary and greyscale format conversion, and more. Even though high-level data and apparent contextual data have not been approved, the method shows promise in learning every image by combining various scale and multiple level visual reactions.

Refrences

[1] Acharya, U. R., Fernandes, S. L., WeiKoh, J. E., Ciaccio, E. J., Fabell, M. K. M., Tanik, U. J., ... & Yeong, C. H. (2019). Automated detection of Alzheimer’s disease using brain MRI images–a study with various feature extraction techniques. Journal of Medical Systems, 43(9), 1-14..

[2] Alorf A, Khan MU. Multi-label classification of Alzheimer's disease stages from resting-state fMRI-based correlation connectivity data and deep learning. Computers in Biology and Medicine. 2022;151:106240.

[3] Bertasius G, Shi J, Torresani L. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 4380-4389).

[4] Bobyr MV, Milostnaya NA, Bulatnikov VA. The fuzzy filter based on the method of areas’ ratio. Applied Soft Computing. 2022;117:108449.

[5] Bobyr MV, Milostnaya NA, Bulatnikov VA. The fuzzy filter based on the method of areas’ ratio. Applied Soft Computing. 2022;117:108449.

[6] Buvaneswari PR, Gayathri R. Deep learning-based segmentation in classification of Alzheimer’s disease. Arabian Journal for Science and Engineering. 2021;46:5373-83.

[7] Diogo VS, Ferreira HA, Prata D, Alzheimer’s Disease Neuroimaging Initiative. Early diagnosis of Alzheimer’s disease using machine learning: a multi-diagnostic, generalizable approach. Alzheimer's Research & Therapy. 2022;14(1):107.

[8] DUBEY S. Alzheimer’s Dataset (4 class of Images) Images of MRI Segementation.

[9] Fuse, H., Oishi, K., Maikusa, N., Fukami, T., & Japanese Alzheimer's Disease Neuroimaging Initiative. (2018, December). Detection of Alzheimer's Disease with Shape Analysis of MRI Images. In 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS) (pp. 1031-1034). IEEE.

[10] Hwang JJ, Liu TL. Pixel-wise deep learning for contour detection. arXiv preprint arXiv:1504.01989. 2015.

[11] Islam, J., Zhang, Y., & Alzheimer’s Disease Neuroimaging Initiative. (2018, December). Deep convolutional neural networks for automated diagnosis of Alzheimer’s disease and mild cognitive impairment using 3D brain MRI. In International Conference on Brain Informatics (pp. 359-369). Springer, Cham.

[12] Koh, J. E. W., Jahmunah, V., Pham, T. H., Oh, S. L., Ciaccio, E. J., Acharya, U. R., ... & Ramli, N. (2020). Automated detection of Alzheimer's disease using bi-directional empirical model decomposition. Pattern Recognition Letters, 135, 106-113.

[13] Kundaram SS, Pathak KC. Deep learning-based Alzheimer disease detection. InProceedings of the Fourth International Conference on Microelectronics, Computing and Communication Systems: MCCS 2019 2021 (pp. 587-597). Springer Singapore.

[14] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 3431-3440).

[15] Mamun M, Shawkat SB, Ahammed MS, Uddin MM, Mahmud MI, Islam AM. Deep Learning Based Model for Alzheimer's Disease Detection Using Brain MRI Images. In2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) 2022 (pp. 0510-0516). IEEE.

[16] Marwa EG, Moustafa HE, Khalifa F, Khater H, AbdElhalim E. An MRI-based deep learning approach for accurate detection of Alzheimer’s disease. Alexandria Engineering Journal. 2023;63:211-21.

[17] Murugan S, Venkatesan C, Sumithra MG, Gao XZ, Elakkiya B, Akila M, Manoharan S. DEMNET: a deep learning model for early diagnosis of Alzheimer diseases and dementia from MR images. IEEE Access. 2021;9:90319-29.

[18] Nawaz H, Maqsood M, Afzal S, Aadil F, Mehmood I, Rho S. A deep feature-based real-time system for Alzheimer disease stage detection. Multimedia Tools and Applications. 2021;80:35789-807.

[19] Rajesh P, Shajin FH, Kannayeram G. A novel intelligent technique for energy management in smart home using internet of things. Applied Soft Computing. 2022; 128:109442.

[20] Rajesh P, Shajin FH, Kumaran GK. An Efficient IWOLRS Control Technique of Brushless DC Motor for Torque Ripple Minimization. Applied Science and Engineering Progress. 2022; 15(3):5514-.

[21] Ren X. Multi-scale improves boundary detection in natural images. InComputer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10 2008 (pp. 533-545). Springer Berlin Heidelberg.

[22] Saridakis KM, Dentsoras AJ. Soft computing in engineering design–A review. Advanced Engineering Informatics. 2008;22(2):202-21.

[23] Savaş S. Detecting the stages of Alzheimer's disease with pre-trained deep learning architectures. Arabian Journal for Science and Engineering. 2022;47(2):2201-18.

[24] Senthilkumaran N, Rajesh R. Image segmentation-a survey of soft computing approaches. In2009 International Conference on Advances in Recent Technologies in Communication and Computing 2009 (pp. 844-846). IEEE.

[25] Shajin FH, Rajesh P, Nagoji Rao VK. Efficient Framework for Brain Tumour Classification using Hierarchical Deep Learning Neural Network Classifier. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization. 2022:1-8.

[26] Shajin FH, Rajesh P, Raja MR. An efficient VLSI architecture for fast motion estimation exploiting zero motion prejudgment technique and a new quadrant-based search algorithm in HEVC. Circuits, Systems, and Signal Processing. 2022; 41(3):1751-74.

[27] Tian J, Smith G, Guo H, Liu B, Pan Z, Wang Z, Xiong S, Fang R. Modular machine learning for Alzheimer's disease classification from retinal vasculature. Scientific Reports. 2021;11(1):1-1.

[28] Solano-Rojas, B., & Villalón-Fonseca, R. (2021). A Low-Cost Three-Dimensional DenseNet Neural Network for Alzheimer's Disease Early Discovery. Sensors, 21(4), 1302.

[29] Witkin A. Scale-space filtering: A new approach to multi-scale description. InICASSP'84. IEEE International Conference on Acoustics, Speech, and Signal Processing 1984 (Vol. 9, pp. 150-153). IEEE.

[30] Yuille AL, Poggio TA. Scaling theorems for zero crossings. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986;(1):15-25.

[31] Nagarathna, C. R., & Kusuma, M. (2022). Automatic Diagnosis of Alzheimer's disease using Hybrid Model and CNN. Journal of Soft Computing Paradigm, 3(4), 322-335.