Advances in Electrical and Computer Engineering **Volume 20, Number 3, 2020**

Diagnosis of Alzheimer's Disease from Brain Magnetic Resonance Imaging Images using Deep Learning Algorithms

Ravi Chandaran SUGANTHE[1], Rukmani Sevalaiappan LATHA[1], Muthusamy GEETHA[1],
Gobichettipalayam Ramakrishnan SREEKANTH[2]
[1]*Department of Computer Science and Engineering, Kongu Engineering College, Tamilnadu, 638060, India*
[2]*Department of Computer Technology(PG), Kongu Engineering College, Tamilnadu, 638060, India*
suganthe_rc@kongu.ac.in

***Abstract*—Alzheimer's disease is one amongst the progressive disorder that cruelly affects the brain cells. It causes the death of nerve cells and tissue loss in brain. It usually tends to start slowly and aggravates overtime. The symptoms of Alzheimer's disease vary from person to person depending on the severity of the unhealthiness. It exhibits behavioral symptoms such as communication impairments, memory loss, taking a longer time to complete usual activities, and change in attitude and behavior. If the problem worsens over time, then it cannot be cured. Hence it should be identified at the earlier stage itself and treat the patient to lead a normal life on their own. Deep learning algorithms exhibit marvelous performance over conventional machine learning algorithms in identifying the complex patterns in the large volumes of high-dimensional medical imaging data. Hence, recently significant attention has been paid to apply deep learning for medical diagnosis. In this research, Deep Convolution Neural Network (DCNN) and VGG-16 inspired CNN (VCNN) models have been built to classify the different stages of Alzheimer's Disease from the Magnetic Resonance Imaging(MRI) images. Experiments are carried out on an ADNI dataset and the results obtained show that the proposed models achieved excellent accuracy.**

***Index Terms*—artificial intelligence, artificial neural network, image classification, machine learning, medical diagnosis.**

I. INTRODUCTION

Alzheimer's disease is a most quickly developing disorder in older people, which generally starts slowly and affects the human brain resulting in the death of brain cells. The individuals affected by Alzheimer's disease gradually lose their ability to think, and they forget their daily activities. Sometimes they are often unable to do their regular activities and take more time to recall their close relative names too. The stages of getting into Alzheimer's disease are cognitive normal (CN), mild cognitive impairment (MCI), and Alzheimer's disease dementia (AD). The early and later stages of MCI are called Early Mild Cognitive Impairment (EMCI) and Late Mild Cognitive Impairment (LMCI), respectively. The MCI is a type of memory loss in which happens before AD or other dementia. If the person frequently forget about their regular activities, this could mean the person may be affected by MCI. About 10 to 20 percent of people over the age of 65 might have MCI. As the National Institute on Aging has suggested, approximately 8 out of 10 individuals who are known to have amnestic MCI will begin to build up Alzheimer's over seven years. AD can be best treated with regular exercise, intake of nutritious food, and spending time with friends and family. AD is the most common form of dementia that causes thinking, memory and behavioral problems. Someone with advanced AD may unable to live on their own and may have anxiety, feel uncertainty, and be unable to contact others in the society. The likelihood that the patient will recover from extreme AD is less and hence, it needs to be predicted and treated in its early stages.

A number of prediction models using machine learning algorithms have been developed in the literature to diagnose Alzheimer's disease from the MRI images. The author in [1] classified the different stages of Alzheimer's disease using Support Vector Machine (SVM) by extracting the most relevant high- level features from the MRI images. However, in any machine learning algorithm, feature extraction prevails to be a difficult task. The SVM algorithm is complex in nature for processing and extracting the image attributes, as well as time-consuming and computationally intensive. Authors in [2-4] used a combination of k-Means, Random Forest, and Region Growing algorithms for the prediction. The k-Means algorithm was applied to cluster the MRI images. From the clustered images, the white and grey matter were extracted using the region Growing algorithm. With the extracted features, Random Forest algorithm was applied to classify the disease with and without neuro-anatomical constraints.

The author in [5] proposed a deep learning algorithm using a stacked auto-encoder and softmax output layer to detect AD which addresses the limitation of the machine learning algorithm. This proposed system detected the AD and MCI stages of the disease. In [6], the author studied the performance of five evolutionary optimization algorithms such as Particle Swarm Optimization, Pattern Search, Bat Algorithm, Simulated Annealing, and Genetic Algorithm in deriving the most suitable features from the 3D MRI images for Alzheimer's disease prediction. These algorithms were used to obtain near-optimum solutions to large scale optimization problems.

Several Machine learning algorithms were used in [7-17] to classify the Alzheimer's disease from he brain MRI images. The classification of Alzheimer's disease was performed by using simple convolutional neural network model with less number of samples in [18-25] and the model

---

Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database.

.

Digital Object Identifier 10.4316/AECE.2020.03007

achieved moderate accuracy.

The authors in [26] performed a Fuzzy Medical Image Retrieval (FMIR) for cancer prediction by combining Vector Quantization with fuzzy signatures and fuzzy s-trees. The performance metrices like modeling error, the mean square error and the percent relative modeling error are analyzed with fuzzy based optimization models in [27-28]. The robotic teleoperation is a successful means of accessing risky locations for offering telepresence. The authors in [29] presented a method for handling huge delays that occur in real-time during teleoperation of a remote surgical robot.

The proper working of the machine learning algorithms rely on the domain expert in extracting the high-level features by reducing the data complexity and making the extracted features more accessible to the learning algorithms. In traditional machine learning algorithms, highly relevant features influence the performance of the diagnostic system. However, the selection of more appropriate features from high complexity data is very difficult and time consuming for the machine learning algorithms. But, deep learning algorithm selects more appropriate features automatically without the domain expertise irrelevant to complexity in the data. This feature motivated us to use deep learning for the diagnosis of Alzheimer’s disease from the MRI images.

In [30-31], the authors made use of an image processing tool named FMRIB Software Library (FSL) for preprocessing the structural MRI images and deep convolutional neural networks for classification. In the preprocessing step, various operaions such as skull stripping, brain tissue segmentation and bias field correction were carriedout using Brain Extraction Tool(BET) and FMRIB's Automated Segmentation Tool(FAST) that are available in FSL.

The rest of the paper is structured as follows: section II offers an overview of the dataset, the basic CNN architecture and the steps involved in the proposed model, sections III and IV describe the DCNN and VCNN classification models for the diagnosis of Alzheimer's disease, section V compares the performance of DCNN and VCNN, and section VI presents the conclusion and future work.

II. MATERIALS AND METHODOLOGY

The dataset used in this experiment is obtained from he publicly available ADNI database. The database contains information about the different views of the brain such as axial view, coronal view and sagittal view. Doctors make use of these images to check whether the patients are in normal condition or affected by any disease, and utilize it to plan for the treatment, since the images indicate the seriousness of the illness.

The proposed classification model makes use of 1000 sample brain images which include the 250 CN images, 250 EMCI images, 250 LMCI and 250 AD images. These images are used for training both DCNN and VCNN models for binary classification of different AD stages. T1 w MRI input images are used as samples. Table I contains the detailed level of participants taken for each classification. 70% of image samples are taken for training and 30% of image samples are considered for testing.

TABLE I. DETAILS OF SUBJECTS TAKEN FOR EXPERIMENT

AD stage	Number of Samples	Age range	No.of Male/Female samples
CN	250	40 to 88	123/127
EMCI	250	48 to 94	131/119
LMCI	250	44 to 89	118/132
AD	250	50 to 96	148/102

The proposed work defines 6 different classification models for different stages in AD using CNN and VGG16 architecture:

1. CN vs EMCI - This model classifies whether the patient is in a cognitive normal stage or enters into an early stage of mild cognitive impairment.
2. AD vs CN – This model classifies whether the patient is in cognitive normal or affected by AD dementia.
3. CN vs LMCI - This model classifies whether the patient is in a cognitive normal stage or enters into severe mild cognitive impairment.
4. AD vs EMCI - This model classifies whether the patient is in AD dementia stage or an early stage of mild cognitive impairment.
5. AD vs LMCI - This model classifies whether the patient is in AD dementia stage or the late stage of mild cognitive impairment.
6. LMCI vs EMCI - This model classifies whether the patient in an early or severe stage of mild cognitive impairment.

Algorithm 1 summarizes the steps for implementing these classification models. The basic functionality of both DCNN and VCNN follow convolutional neural network model.

Algorithm 1. Alzheimer’s Disease Classification Model

1. Prepare the dataset
    - 1.1 Preprocess the dataset images for skull stripping, brain tissue segmentation and bias field correction operations using FSL tool.
    - 1.2 Set the input shape based on image_data_format (channel first or channel last)
    - 1.3 Scale down the pixel values from the the range 0 to 255 to the range 0 to 1
    - 1.4 Split the dataset into training set (70%) and testing set(30%)
2. Build the Classification Model (DCNN/VCNN) by setting the activation function as Relu in all the hidden layers and Sigmoid/Softmax function in the output layer.
3. Train the model using training dataset
4. Validate the classification model using testing dataset
5. Repeat training and validation phases for required number of epochs to get high accuracy and minimum loss.
6. Test the diagonsis of the alzheimers’ disease by applying unseen images as input.
7. If accuracy is less and/or loss is more, repeat training and testing process considering the following design choices:
    - Convolution window size – Train the model with different filter sizes such as 3x3, 5x5 etc., and different number of filters.
    - Striding – reduce the exection time and the size of the feature map by applying striding
    - Padding - In the input image, if any important information is present in the corner of the image apply padding.
    - Pooling- Extract the high level features and reduce the input feature map dimension using pooling operation.
    - Dropout –Applied in the input or hidden layers alone to overcome the overfitting
8. Data augmentation – If desired accuracy is not achieved or overfitting is detected, generate more training samples and train the model with these samples. With data augmentation, model with good generalization performance can be achieved.

Functionality of each layer in the basic CNN Model

In each convolution layer of CNN, multiple filters are applied to compute different feature maps. Each neuron of a feature map is computed using some or all the neurons in the previous layer. The output feature map in each node is obtained by first applying the filter on the feature map taken from the previous layer followed by applying a nonlinear activation function. The complete feature maps are obtained by applying different filters. The feature at (i, j) in the k^{th} feature map of L^{th} layer is represented as ($z_{ijk}^{(L)}$) and is computed by using (1).

$$z_{ijk}^{(L)} = (w_k^{(L)})^T * x_{ij}^{(L)} + b_k^{(L)} \quad (1)$$

where $w_k^{(L)}$ and $b_k^{(L)}$ are the weight vector and bias of the kth filter of L-th layer, respectively, and $x_{ij}^{(L)}$ is the input feature at location (i, j) of the L^{th} layer.

After applying the convolution function on the input feature map with filters, the next activation function is applied on $z_{i,j,k}^{(L)}$ to introduce nonlinearity property to the CNN. This nonlinear property is needed to detect the important features in multilayer networks.

The activation value at location (i,j) in k^{th} feature map at layer L is denoted as $a_{ijk}^{(L)}$ which is computed by applying the activation function on $z_{ijk}^{(L)}$. It is denoted as in (2).

$$a_{ijk}^{(L)} = f(z_{ijk}^{(L)}) \quad (2)$$

The familiar activation functions used now-a-days are sigmoid, softmax, tanh and ReLU. Activation functions are selected based on the output required in the model. ReLU is used in all the hidden layers of the proposed CNN model and this function always generates a positive value that lies between 0 and 1. If the input is positive then the output is same as input otherwise the output is zero. After applying the activation function, the output feature map of this layer is given to the pooling layer to reduce the size of the feature map. Feature map of a pooling layer x is derived from a feature map of the preceding convolutional layer present in the position x.

After processing the input sequentially through several convolutions and pooling layers, it is sent to one or more fully connected layers to extract the important features. In a fully connected layer, each and every neuron from the previous layer is connected to each and every neuron of the current layer to generate high-level features. The last layer of CNN model is the output layer which takes sigmoid as its activation function, since it is needed for binary classification. The equation for the sigmoid function is given in (3):

$$f(z) = \frac{1}{(1+e^{-z})} \quad (3)$$

To get the optimum result for the classification of images, the error in the model's current state must be estimated repeatedly using appropriate error/loss function. The loss in the next evaluation of the model is minimized by adjusting the weights based on the loss value.

Suppose there are 'n' samples $\{(x_i, y_i): i = 1,2,3,...\}$, where x_i is the i^{th} input sample and y_i is the target output for the input sample x_i. Let o_i be the output derived from the model for the input x_i. To calculate the loss, several loss functions are used in deep learning algorithms such as Binary Cross-Entropy, Mean Squared Error Loss, Mean Absolute Error Loss, Hinge Loss, Squared Hinge Loss, etc. The proposed model makes use of Mean Square Error loss which is calculated using equation (4) where y is the target value and o is the predicted value of all the n samples. Here yi is the target label and oi is the predicted value of ith sample.

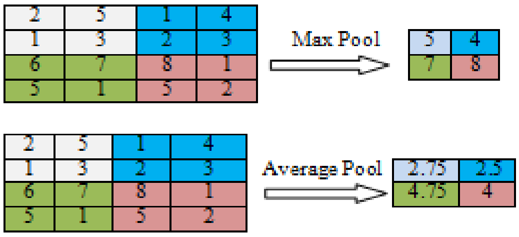
$$L(y,o) = \frac{1}{n}\sum_{i=1}^{n}(y_i - o_i)^2 \quad (4)$$

III. Deep Convolutional Neural Network for Alzheimer's Disease Classification

Convolutional Neural Network (CNN) is a type of artificial neural network which is familiar for the classification of images because of its biggest benefit of feature extraction compared with the machine learning algorithms. In this work, T1 w MRI images are used as input samples. The main objective of the convolutional layer in CNN is to extract the high-level features and store these extracted features as a 'feature map'. This model uses many convolutional layers for tuning to get a more accurate system. The first convolutional layer is most responsible for capturing the edges and image color. With added convolutional layers, the model takes a better understanding of input images. Two kinds of results are produced by the convolutional layers: some of the layers reduce the dimensionality of the input image without affecting the important higher-level features, and the others which expands the dimensionality is either expanded or continues as before.

As the feature map is derived from the input image, the convolution layer records the features of the input image along with a feature location. This is a limitation in a feature map because small changes in feature location generate different feature map for the input image. These changes are the operations that are applied to the input image, such as rotating, cropping, shifting etc. To overcome the above mentioned limitation, the downsampling is applied. Hence, the output of the convolutional layer is given to the pooling layer which is responsible for reducing the size of each feature map i.e., reducing convolved features with down-sampling operation along with a spatial dimension. There are two types of pooling layers available: Max pooling and Average pooling. Average pooling summarizes the average presence of a feature and max pooling adds-up the most active presence of a feature. The Fig.2 shows the process of downsampling with Max pooling and average pooling using a 2x2 filter and stride 2.

The output from the final pooling layer is first flattened and then given as input to the fully connected layer. This fully connected layer performs the final classification.



2	5	1	4
1	3	2	3
6	7	8	1
5	1	5	2

5	4
7	8

2	5	1	4
1	3	2	3
6	7	8	1
5	1	5	2

2.75	2.5
4.75	4

Figure 2. Down-sampling with Max-pooling and Average-pooling

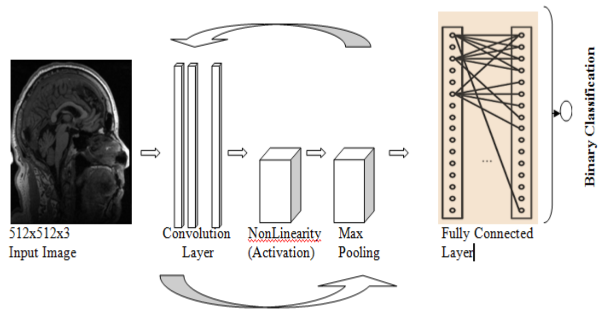


Figure 3. DCNN Framework for Alzheimer’s Disease classification

The DCNN framework for classifying the Alzheimer’s disease is depicted in Fig.3. The DCNN model consists of a varying number of convolutional layers and fully connected layers. All the layers except the output layer make use of the ‘Relu’ activation function. ‘Sigmoid’ function is used bythe outpt layer. The input, output and the number of parameters to be updated in each layer of DCNN model with 4 convolution layers is explained below:

First layer: Each image in the dataset is of size 512x512 RGB (three channels). This input image is sent to the first convolution layer and is processed with 16 convolution filters of size 2 × 2. The number of parameters to be trained in this layer is 2 x 2 x 3 x 16 + 16 bias = 208. The output of this layer is 511 x 511 x 16 since each filter creates an image of size 511x511.

Second Layer: The second convolutional layer uses the input of size 255x255x16 with 32 filters of size 2×2. No.of parameters to be trained in this layer is 2x2x16 x32+32 bias = 2080. The output of this layer is 254 x 254 x 32 since each filter creates an image of size 254x254 and 32 filter creates 32 images of size 254x254.

Third Layer: The third layer uses images with 64 filters having size 2×2. No.of parameters to be trained in this layer is 2x2x32x64+ 64bias = 8256. The output of this layer is 126 x 126 x 64 since each filter creates an image of size 126x126 and 64 filters create 64 images of size 126x126.

Fourth Layer: The fourth layer uses images with 128 filters having size 2×2. No.of parameters to be trained in this layer is 2x2x64 x128+128 bias = 32,896. The output of this layer is 62 x 62 x 128 since each filter creates an image of size 62x62 and 128 filter creates 128 images of size 62x62.

Fifth Layer: This is the max pooling layer to which output from the fourth layer is sent. It generates an output of size 31x31x128 and this 3D data is flattened to generate a vector of size 1,23,008. This vector is given as input to the fully connected layer which makes use of 64 filters of size 1x1 and generates the output of size 1x1x64=64. Number of parameters to be updated in this layer is 123008x64+64 bias=78,72,576.

Output Layer: Finally, there is a fully connected layer that uses the sigmoid activation function with 2 possible values. This layer takes input from a fully connected layer and computes the class scores and gives the binary classification. The number of parameters updated in this layer is 64x1+1 bias=65. This classification shows whether the subject is affected by CN/EMCI/LMCI/AD.

IV. RESULTS AND DISCUSSION OF DCNN

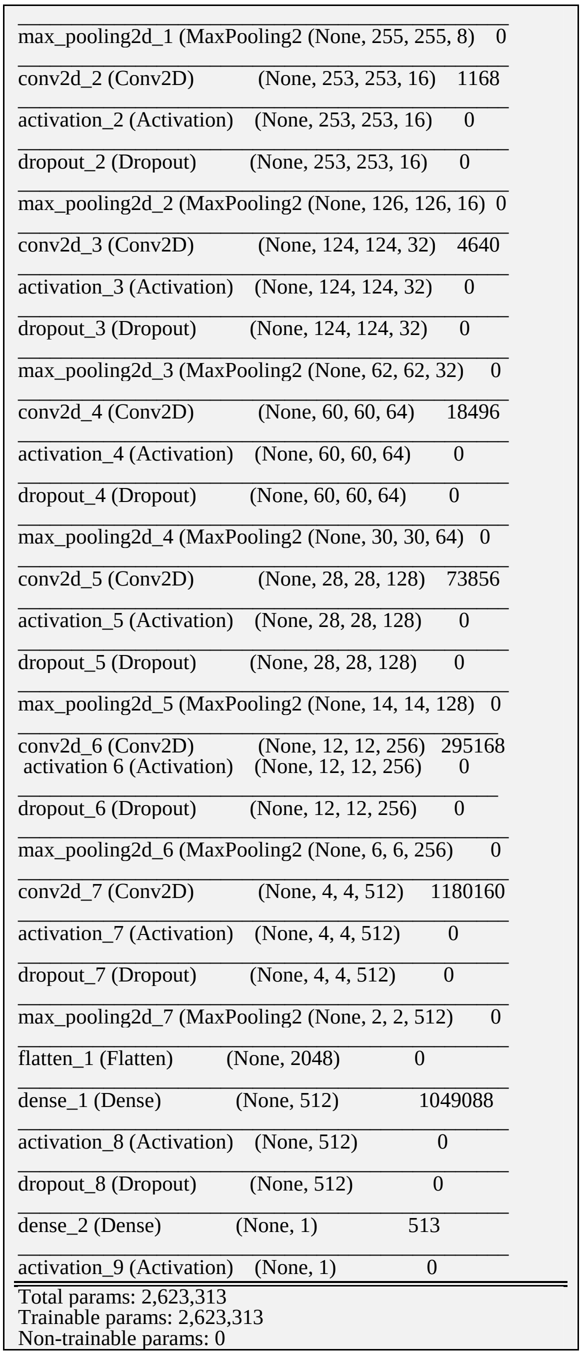
In the proposed DCNN model, structural MRI images from ADNI dataset are used for training the model. Six different DCNN models are trained for classifying CN vs EMCI, AD vs CN, CN vs LMCI, AD vs EMCI, AD vs LMCI and EMCI vs LMCI. The model is applied by varying the number of convolution layers, filter size, number of filters, dropout layers and pooling layers with different pool window size. With these variations, the model which gave the highest accuracy has been selected. The DCNN model with 7 convolution layers, 3x3 convolution filter, 2x2 pool window, Mean Square Error loss, Stochastic gradient descent optimizer gave the highest accuracy in 15 epochs. Table II and III show the summary of the different layers and the learnable parameters used in DCNN having 4 convolution layers and 7 convolution layers, respectively. The parameters used in DCNN with 7 convolution layer is given in Table IV.

TABLE II. SUMMARY OF LAYERS AND PARAMETERS IN DCNN WITH 4 CONV. LAYERS

```
Model: "sequential_1"
_________________________________________________________
Layer (type)              Output Shape            Param #
=========================================================
conv2d_1 (Conv2D)          (None, 511, 511, 16)     208
_________________________________________________________
activation_1 (Activation)  (None, 511, 511, 16)     0
_________________________________________________________
dropout_1 (Dropout)        (None, 511, 511, 16)     0
_________________________________________________________
max_pooling2d_1 (MaxPooling2(None, 255, 255, 16)  0
_________________________________________________________
conv2d_2 (Conv2D)          (None, 254, 254, 32)  2080
_________________________________________________________
activation_2 (Activation)  (None, 254, 254, 32)     0
_________________________________________________________
dropout_2 (Dropout)        (None, 254, 254, 32)     0
_________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 127, 127, 32)  0
_________________________________________________________
conv2d_3 (Conv2D)          (None, 126, 126, 64) 8256
_________________________________________________________
activation_3 (Activation)  (None, 126, 126, 64)     0
_________________________________________________________
dropout_3 (Dropout)        (None, 126, 126, 64)     0
_________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 63, 63, 64)  0
_________________________________________________________
conv2d_4 (Conv2D)       (None, 62, 62, 128)    32896
_________________________________________________________
activation_4 (Activation)  (None, 62, 62, 128)      0
_________________________________________________________
dropout_4 (Dropout)        (None, 62, 62, 128)      0
_________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 31, 31, 128) 0
_________________________________________________________
flatten_1 (Flatten)        (None, 123008)          0
_________________________________________________________
dense_1 (Dense)            (None, 64)            7872576
_________________________________________________________
activation_5 (Activation)  (None, 64)              0
_________________________________________________________
dropout_5 (Dropout)        (None, 64)              0
_________________________________________________________
dense_2 (Dense)            (None, 1)              65
_________________________________________________________
activation_6 (Activation)  (None, 1)               0
=========================================================
Total params: 7,916,081
Trainable params: 7,916,081
Non-trainable params: 0
```

TABLE III. SUMMARY OF LAYERS AND PARAMETERS IN DCNN WITH 7 CONV. LAYERS

```
Model: "sequential_1"
_________________________________________________________
Layer (type)              Output Shape            Param #
=========================================================
conv2d_1 (Conv2D)          (None, 510, 510, 8)      224
_________________________________________________________
activation_1 (Activation)  (None, 510, 510, 8)      0
_________________________________________________________
dropout_1 (Dropout)        (None, 510, 510, 8)      0
```



```
max_pooling2d_1 (MaxPooling2 (None, 255, 255, 8)   0
conv2d_2 (Conv2D)           (None, 253, 253, 16)   1168
activation_2 (Activation)   (None, 253, 253, 16)   0
dropout_2 (Dropout)         (None, 253, 253, 16)   0
max_pooling2d_2 (MaxPooling2 (None, 126, 126, 16)  0
conv2d_3 (Conv2D)           (None, 124, 124, 32)   4640
activation_3 (Activation)   (None, 124, 124, 32)   0
dropout_3 (Dropout)         (None, 124, 124, 32)   0
max_pooling2d_3 (MaxPooling2 (None, 62, 62, 32)    0
conv2d_4 (Conv2D)           (None, 60, 60, 64)     18496
activation_4 (Activation)   (None, 60, 60, 64)     0
dropout_4 (Dropout)         (None, 60, 60, 64)     0
max_pooling2d_4 (MaxPooling2 (None, 30, 30, 64)    0
conv2d_5 (Conv2D)           (None, 28, 28, 128)    73856
activation_5 (Activation)   (None, 28, 28, 128)    0
dropout_5 (Dropout)         (None, 28, 28, 128)    0
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 128)   0
conv2d_6 (Conv2D)           (None, 12, 12, 256)    295168
 activation 6 (Activation)  (None, 12, 12, 256)    0
dropout_6 (Dropout)         (None, 12, 12, 256)    0
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 256)     0
conv2d_7 (Conv2D)           (None, 4, 4, 512)      1180160
activation_7 (Activation)   (None, 4, 4, 512)      0
dropout_7 (Dropout)         (None, 4, 4, 512)      0
max_pooling2d_7 (MaxPooling2 (None, 2, 2, 512)     0
flatten_1 (Flatten)         (None, 2048)           0
dense_1 (Dense)             (None, 512)            1049088
activation_8 (Activation)   (None, 512)            0
dropout_8 (Dropout)         (None, 512)            0
dense_2 (Dense)             (None, 1)              513
activation_9 (Activation)   (None, 1)              0
=================================================================
Total params: 2,623,313
Trainable params: 2,623,313
Non-trainable params: 0
```

TABLE IV. DETAILS OF PARAMETERS USED IN DCNN(7 LAYERS)

Parameters	Parameter Values
No.of Convolution layers	7
Dataset	ADNI standard MRI
Images	T1 w MRI saggital view
Image size	512x512
batch_size	32
Input layer - Activation Function	ReLu
Hidden layer - Activation Function	ReLu
Output layer - Activation Function	Sigmoid
Dropout in hidden layers	20%
Dropout in Dense layer-1	50%

Optimizer	SGD
	Learning rate=0.1
	decay=1e-6,
	momentum=0.9
	nesterov=True

Loss function	mean_squared_error
Class mode	Binary
No.of Epochs	15
steps_per_epoch	400
validation_steps	100

TABLE V. DETAILS OF PARAMETERS USED IN VCNN

Parameters	Parameter Values
No.of Convolution layers	13
Data set	ADNI standard MRI images
Images	T1 w MRI saggital view images
Image size	512x512, 3 Channels RGB image
batch_size	32
Input layer – Activation Function	ReLu
Hidden layer – Activation Function	ReLu
Output layer – Activation Function	Softmax

Optimizer	SGD
	Learning rate=0.1
	decay=1e-6
	momentum=0.9
	nesterov=True

Loss function	mean_squared_error
Class mode	Binary
No.of Epochs	15
Steps_per_epoch	400
Validation_steps	100
callbacks	[checkpoint,early]

Early Stopping	monitor='val_accuracy'
	min_delta=0, patience=20
	verbose=1,mode='auto'

TABLE VI. PERFORMANCE OF DCNN WITH 4 CONV. LAYERS

Classification	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
CN vs EMCI	94.23	4.76	76.41	17.04
AD vs CN	89.21	7.82	80.52	17.16
CN vs LMCI	95.6	3.58	88.64	16.23
AD vs EMCI	89.61	7.27	88.30	12.46
AD vs LMCI	96.27	2.75	90.34	13.76
LMCI vs EMCI	91.37	6.78	80.87	18.56

The performance of DCNN with these two models are given in Table VI and Table VII, respectively and it is found that DCNN with 7 convolution layer works much better than DCNN with 4 convolution layers. Among all the six models, AD vs LMCI classification achieves higher test accuracy of about 93.76% than the other classifications. The accuracy and loss is different in each model, which may be due to the number of input samples as well as the quality of the images used for training and testing. The graphs in Fig.4 shows the performance achieved in 5, 10 and 15 epochs on 6 different binary classifications of DCNN model with 7 convolution layers. These graphs show that accuracy increases and loss decreases when the number of epochs are increased.

TABLE VII. PERFORMANCE OF DCNN WITH 7 CONV. LAYERS

Classification	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
CN vs EMCI	93.81	5.61	93.44	12.79
AD vs CN	93.90	4.58	91.40	3.17
CN vs LMCI	97.53	2.05	93.76	8.68
AD vs EMCI	95.43	3.73	91.56	9.75

AD vs LMCI	97.45	2.02	93.28	8.5
LMCI vs EMCI	93.23	5.14	89.68	10.64

V. VCNN ARCHITECTURE FOR ALZHEIMER’S DISEASE CLASSIFICATION

The VGG16 inspired CNN framework (VCNN) architecture consists of 13 convolution layers, 5 Max Pooling layers, and 3 Dense layers. There are totally 21 layers present in this architecture but the eight has to be updated only for 16 layers. These 21 layers are arranged as five convolution blocks followed by one fully connected classifier block. Each convolution block consists of 2 or 3 convolution layers (Convolution2D) followed by the Pooling layer (MaxPooling2D). RGB image of size 224x224 is given as an input to the first convolution layer of the first block. The last fully connected block firsr flattens the input and then passes to it to the fully connected layer. All the convolution layers except the last one use ReLU as its activation function. The last convolution layer uses Softmax activation function for the classification of images.

Fig.5 illustrates the VCNN framework for Alzheimer’s disease classification between two Alzheimer’s stages. First, the RGB images of size 512x512 are converted into a 224x224 images and then these images are given as input to the VCNN model. The input image size, output size, number and size of filters used in each layer are given below:

First and Second Layers: The input for this architecture is 224x224 RGB image which passes through first and second convolution layers with 64 filters having a window size of 3×3 and same pooling size with a stride of 1. The size of the image changes to 224x224x64. Then it is applied to MaxPool layer with a filter size 3×3 and a stride of 2. The output image is reduced to 112x112x64 and a dropout of 20% is used.

Third and Fourth Layer: Followed by first and second convolution layers, there are two convolution layers with 128 filters each of size 3×3 and a stride 1. Then there is a MaxPool layer with its filter size 3×3 and a stride of 2. This layer has 128 filters which reduces the output to 56x56x128. The dropout used in these layers is 30%.

Eighth to Thirteenth Layers: There are two sets of 3 convolution layers each followed by a MaxPool layer. All convolution layers have 512 filters each of size 3×3 and a stride of one. The final image size after the last MaxPool layer will be reduced to 7x7x512.

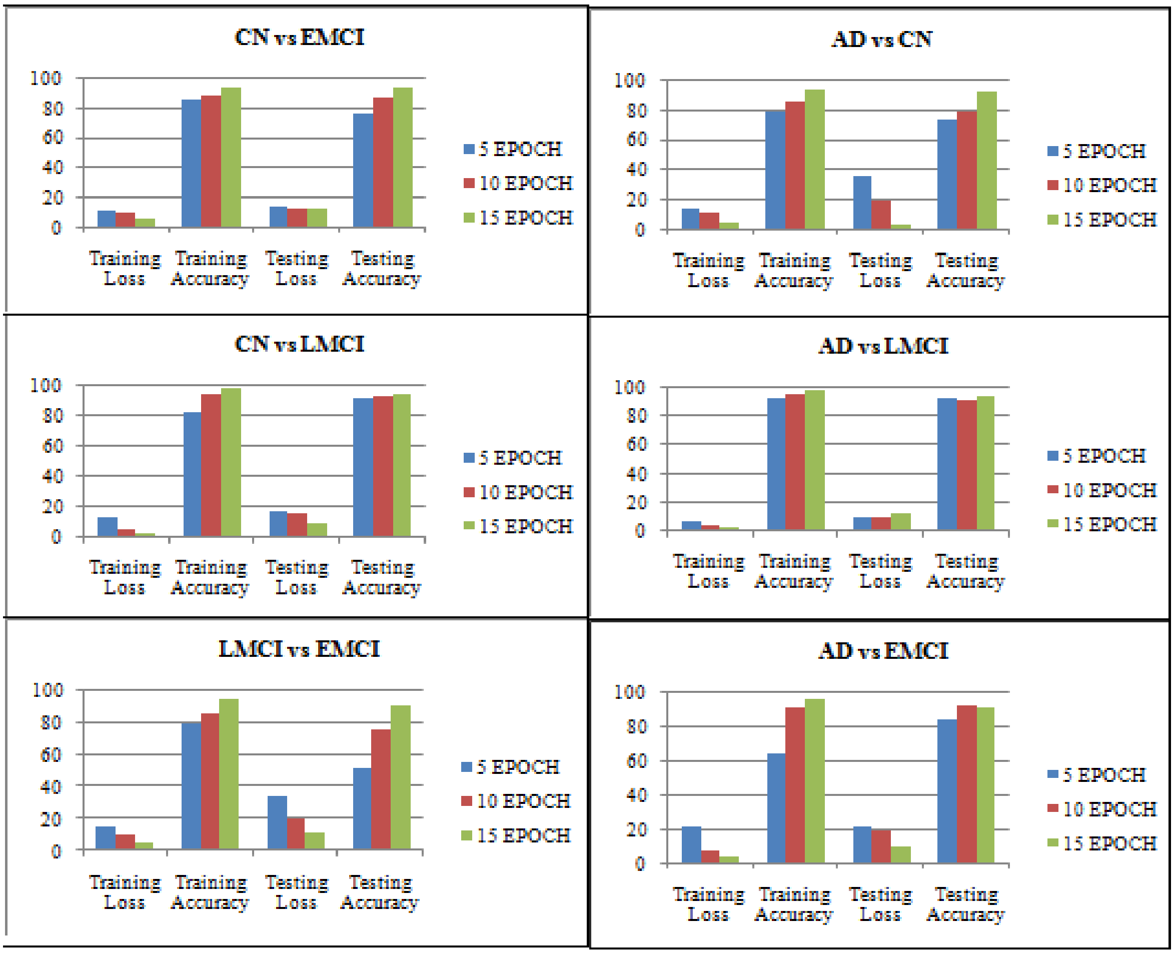


Figure 4. Performance of DCNN with 7 Conv. Layers

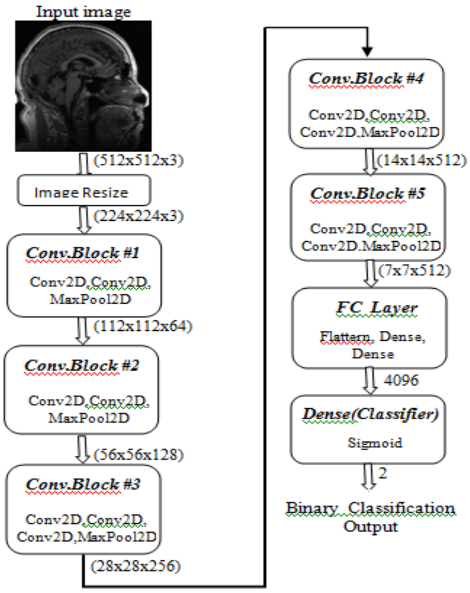


Figure 5. VCNN Framework for Alzheimer's Disease classification

Fourteenth Layer: The output from the convolution layer is of size 7x7x512 which is flattened by passing it through a fully connected layer. The output obtained from this layer is of size 25088.

Fifteenth and Sixteenth Layer: Output from the above layer is fed as input to these two fully connected layers. These layers then process the input and produces an output consisting of 4096 features.

Output Layer: This is a softmx layer. With the 4096 features from the previous two layers, this layer generates 2 classified outputs.

Summary of the layers and the parameters present in VCNN model is given in Fig.6**.**

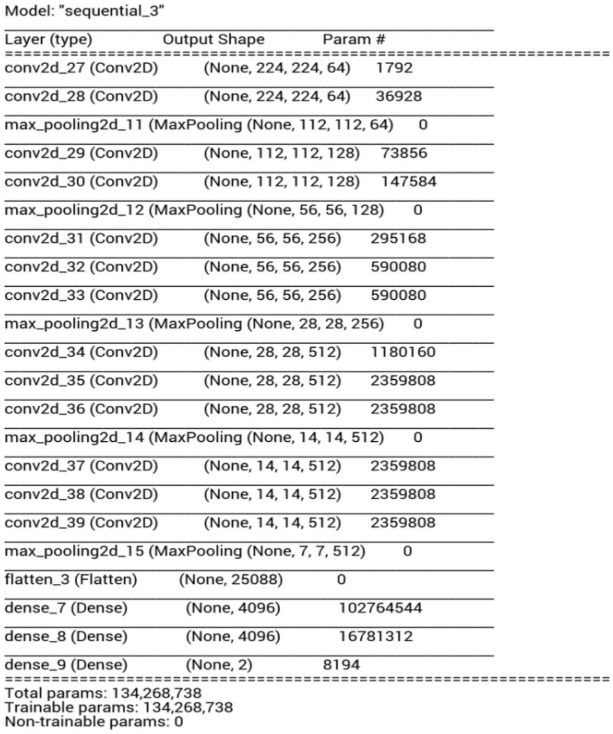

```
Model: "sequential_3"
________________________________________________________________
Layer (type)               Output Shape              Param #
================================================================
conv2d_27 (Conv2D)         (None, 224, 224, 64)      1792
________________________________________________________________
conv2d_28 (Conv2D)         (None, 224, 224, 64)      36928
________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 112, 112, 64)    0
________________________________________________________________
conv2d_29 (Conv2D)         (None, 112, 112, 128)     73856
________________________________________________________________
conv2d_30 (Conv2D)         (None, 112, 112, 128)     147584
________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 56, 56, 128)     0
________________________________________________________________
conv2d_31 (Conv2D)         (None, 56, 56, 256)       295168
________________________________________________________________
conv2d_32 (Conv2D)         (None, 56, 56, 256)       590080
________________________________________________________________
conv2d_33 (Conv2D)         (None, 56, 56, 256)       590080
________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 28, 28, 256)     0
________________________________________________________________
conv2d_34 (Conv2D)         (None, 28, 28, 512)       1180160
________________________________________________________________
conv2d_35 (Conv2D)         (None, 28, 28, 512)       2359808
________________________________________________________________
conv2d_36 (Conv2D)         (None, 28, 28, 512)       2359808
________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 14, 14, 512)     0
________________________________________________________________
conv2d_37 (Conv2D)         (None, 14, 14, 512)       2359808
________________________________________________________________
conv2d_38 (Conv2D)         (None, 14, 14, 512)       2359808
________________________________________________________________
conv2d_39 (Conv2D)         (None, 14, 14, 512)       2359808
________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 7, 7, 512)       0
________________________________________________________________
flatten_3 (Flatten)        (None, 25088)             0
________________________________________________________________
dense_7 (Dense)            (None, 4096)              102764544
________________________________________________________________
dense_8 (Dense)            (None, 4096)              16781312
________________________________________________________________
dense_9 (Dense)            (None, 2)                 8194
================================================================
Total params: 134,268,738
Trainable params: 134,268,738
Non-trainable params: 0
```

Figure 6. Summary of layers and parameters in VCNN

VI. RESULTS AND DISCUSSION OF VCNN

VCNN is used for the classification of various AD stages. CN vs EMCI classification achieves higher accuracy of about 96.19% when compared to the other classifications. Performance achieved in terms of accuracy and loss by VCNN for 6 different binary classifications are mentioned in the Table VIII.

TABLE VIII. PERFORMANCE OF VCNN FOR 6 DIFFERENT CLASSIFICATIONS

Classification	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
CN vs EMCI	95.60	3.93	96.19	6.78
AD vs CN	94.12	4.80	93.78	4.82
CN vs LMCI	98.67	4.70	94.87	5.18
AD vs EMCI	94.90	5.90	92.80	6.70
AD vs LMCI	97.80	1.74	94.79	6.67
LMCI vs EMCI	94.70	5.61	92.91	9.79

VII. PERFORMANCE ANALYSIS OF DCNN AND VCNN

We have analyzed the performance of the proposed system by considering the parameters such as training accuracy, training loss, testing accuracy and testing loss. Accuracy is represented as how close the computed classified value to the correct labeled value. It also ensures success of the model. The loss value represents how poorly the system works. If the loss is greater than 15%, then the model needs to be optimized by adjusting the number of convolution layers, the number of filters in each layer, pooling operation, stride and dropout layers. To reduce the loss and improve the accuracy, the model can be trained with more input samples.

The graph shown in Fig.7 depicts the performance in terms of accuracy achieved through the DCNN model and VCNN model. Even though the accuracy produced by the VCNN model is higher than DCNN model, the number of trainable parameters in the VCNN model is more than the DCNN model. To validate these models, you can use the code available on the link: https://drive.google.com/file/d/1015l9FVmTyhbetPJisxrtHMBIPzm_Ivh/view?usp=sharing

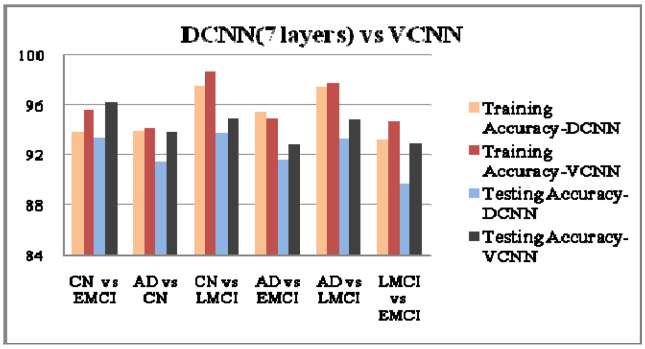


Figure 7. Performance comparison of DCNN (7 layers) and VCNN

VIII. CONCLUSION

The proposed system has successfully classified the AD and their stages using DCNN and VCNN models. These models are trained and validated using 1000 MRI images with their sagittal view. More than 90% of accuracy is achieved in both the models. Among these two models, VCNN produces higher accuracy. A more accurate model can be designed with the proper selection of the number of layers, the number of filters, filter size, Pooling layers, Dropout layers, etc. by keeping the size of the model as minimum. DCNN model can be further improved by applying more number of samples from different datasets. This study mainly focused on the sagittal view of the T1 weighted MRI images. As a future enhancement, other views such as coronal and axial views also will be considered. Accuracy, response time and loss can be further improved by making use of other deep learning algorithms.

REFERENCES

[1] C. Stolojescu-Crisan, S. Holban, "A comparison of x-ray image segmentation techniques," Advances in Electrical and Computer Engineering, vol.13, no.3, pp.85-92, 2013, doi:10.4316/AECE.2013.03014.

[2] A. V. Lebedev, E. Westman, G. J. P. Van Westen, M. G. Kramberger, A. Lundervold, D. Larsland, et al., ”Random Forest ensembles for detection and prediction of Alzheimer's disease with a good between-cohort robustness”, NeuroImage: Clinical Journal, vol. 6, pp. 115-125, 2014, doi: 10.1016/j.nicl.2014.08.023.

[3] U. R. Acharya, S. L. Fernandes, J. E. Wei Koh, E. J. Ciaccio, M. K. M. Fabell, U. J. Tanik et al. ,“Automated detection of Alzheimer’s disease using brain MRI images– A study with various feature extraction techniques” Journal of Medical Systems, vol.43, no.9, pp.1-8,2019, doi:10.1007/s10916-019-1428-9.

[4] Mohamed M. Dessouky and Mohamed A. Elrashidy ”Feature extraction of the Alzheimer’s disease images using different optimization algorithms,” Journal of Alzheimer’s Disease & Parkinsonism , vol.6, no.2, pp.1-11, 2016, doi:10.4172/2161-0460.1000230.

[5] S. Liu, S. Liu, W. Cai, S. Pujol, R. Kikinis and D. Feng, "Early diagnosis of alzheimer's disease with deep learning," 2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI), Beijing,July 31, pp. 1015-1018, 2014, doi: 10.1109/ISBI.2014.6868045.

[6] J. Dukart, K. Mueller, H. Barthel, A. Villringer, O. Sabri, M. L. Schroeter,”Meta-analysis based SVM classification enables accurate detection of alzheimer’s disease across different clinical centers using FDG-PET and MRI,” Psychiatry research, vol. 212, no.3, pp. 230-236, 2013, doi: 10.1016/j.pscychresns.2012.04.007.

[7] K. Ota, N. Oishi, K. Ito, H. Fukuyama , J. Sead, Study Group & Alzheimer's Disease Neuroimaging Initiative, “Effects of imaging modalities, brain atlases and feature selection on prediction of Alzheimer's disease,” Journal of Neurosci Methods, vol.256, pp.168-183,2015, doi:10.1016/j.jneumeth.2015.08.020.

[8] F. Malik, S. Farhan, M. A. Fahiem, "An ensemble of classifiers based approach for prediction of Alzheimer's disease using fMRI images based on fusion of volumetric, textural and hemodynamic features," Advances in Electrical and Computer Engineering, vol.18, no.1, pp.61-70, 2018, doi:10.4316/AECE.2018.01008.

[9] A. Ayub, S. Farhan, M. A. Fahiem, H. Tauseef, "A novel approach for the prediction of conversion from mild cognitive impairment to Alzheimer's disease using MRI images," Advances in Electrical and Computer Engineering, vol.17, no.2, pp.113-122, 2017, doi:10.4316/AECE.2017.02015.

[10] O. Geman, C. O. Turcu, A. Graur, "Parkinson's disease assessment using fuzzy expert system and nonlinear dynamics," Advances in Electrical and Computer Engineering, vol.13, no.1, pp.41-46, 2013, doi:10.4316/AECE.2013.01007.

[11] J. Weller, & A. Budson, “Current understanding of Alzheimer’s disease diagnosis and treatment,” F1000 Research, 7,2018, doi: 10.12688/f1000research.14506.1. PMID: 30135715; PMCID: PMC6073093.

[12] M. W. Bondi, E. C. Edmonds, Salmon DP. "Alzheimer's disease: past, present, and future”, J Int Neuropsychol Soc, vol.23(9-10), pp.818-831, 2017, doi:10.1017/S135561771700100X.

[13] B. Dubois, A. Padovani, P.Scheltens, A. Rossi & G.Dell’Agnello, “Timely diagnosis for alzheimer’s disease: A literature review on benefits and challenges”, Journal of Alzheimer's Disease, vol. 49, no. 3, pp. 617-631, 2016 doi: 10.3233/JAD-150692.

[14] M.Tanveer, B.Richhariya, R.U.Khan, A.H.Rashid, P.Khanna, M.Prasad et al., “Machine learning techniques for the diagnosis of Alzheimer’s disease: A review,” ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), vol.16,no.1S, pp.1-35, 2020, doi:10.1145/3344998.

[15] C.K.Fisher, A.M.Smith, J.R.Walsh,“Machine learning for comprehensive forecasting of Alzheimer’s disease progression,” Science Reports 9, vol.13622, 2019, doi:10.1038/s41598-019-49656-2.

[16] J. De Velasco Oriol, E. E. Vallejo, K. Estrada, G. Taméz Peña and ADNI, “Benchmarking machine learning models for late-onset alzheimer’s disease prediction from genomic data,” BMC Bioinformatics 20, vol.709, 2019, doi:10.1186/s12859-019-3158-x.

[17] L. R. Trambaiolli, A. C. Lorena, F. J. Fraga, , P. A. M. Kanda, R. Anghinah & R. Nitrini, “Improving alzheimer’s disease diagnosis with machine learning techniques,” Clinical EEG and Neuroscience, vol.42, no.3, pp.160–165, 2011, doi:10.1177/155005941104200304.

[18] S. Basaia, F. Agosta, L. Wagner, E. Canu, G. Magnani, R. Santangelo et al., and Alzheimer's Disease Neuroimaging Initiative, “Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks,” NeuroImage: Clinical Journal,vol.21,no.1, pp.1-8, 2019, doi:10.1016/j.nicl.2018.101645.

[19] O. Geman, H. Costin, "Automatic assessing of tremor severity using nonlinear dynamics, artificial neural networks and neuro-fuzzy classifier," Advances in Electrical and Computer Engineering, vol.14, no.1, pp.133-138, 2014, doi:10.4316/AECE.2014.01020.

[20] Y. LeCun, Y. Bengio, G. Hinton “Deep learning,” Nature, vol.521, no.1, pp.436–444,2015, doi:10.1038/nature14539.

[21] A. Krizhevsky, I. Sutskever, G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Communication of the ACM, vol.60, no.6, pp.84-90,2017, doi:10.1145/3065386.

[22] N. Sharma, V. Jain & A. Mishra, “An analysis of convolutional neural networks for image classification,” Procedia Computer Science,vol.132, pp.377-384, 2018, doi:10.1016/j.procs.2018.05.198.

[23] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, H. Adeli, D. P. Subha “Automated EEG-based screening of depression using deep convolutional neural network,” Computer Methods and Programs in Biomedicine, vol.161, pp.103–113, 2018, doi:10.1016/j.cmpb.2018.04.012.

[24] S. E. Spasov, L. Passamonti, A. Duggento, P. Lio, N. Toschi , “A multi-modal convolutional neural network framework for the prediction of Alzheimer's disease,” Conf Proc IEEE Eng Med Biol Soc. 2018, pp.1271‐1274, 2018, doi:10.1109/EMBC.2018.8512468.

[25] A. Payan, G. Montana, “Predicting Alzheimer’s disease: A neuroimaging study with 3D convolutional neural networks,” arXiv, February 10, 2015, arXiv:1502.02506.

[26] J. Nowaková, M. Prílepok & V. Snášel, “Medical image retrieval using vector quantization and fuzzy S-tree,” Journal of Medical Systems, vol. 41, no. 18, pp. 1-16, 2017, doi:10.1007/s10916-016-0659-2.

[27] Elena-Lorena Hedrea, Radu-Emil Precup, Claudia-Adina Bojan-Dragos, “Results on tensor product-based model transformation of magnetic levitation systems,” Acta Polytechnica Hungarica, vol. 16, no. 9, pp. 93-111, 2019.

[28] R. P. Gil, Z. C. Johanyák & T. Kovács, “Surrogate model based optimization of traffic lights cycles and green period ratios using microscopic simulation and fuzzy rule interpolation,” International journal of artificial intelligence, vol.16, pp.20-40,2018, Corpus ID: 53310666.

[29] T. Haidegger, L. Kovacs, S. Preitl, R. E. Precup, B. Benyo, Z. Benyo, “Controller design solutions for long distance telesurgical applications,” International Journal of Artificial Intelligence, vol. 6, no.11S , pp. 48-71,2011.

[30] https://fsl.fmrib.ox.ac.uk/fsl/fslwiki

[31] M. Jenkinson, C. Beckmann, T. E. J. Behrens, M. W. Woolrich, S. M. Smith, “FSL,” NeuroImage, vol.62, no.2, pp.782-790, 2012, doi:10.1016/j.neuroimage.2011.09.015.