Multimedia Tools and Applications (2024) 83:3767–3799
https://doi.org/10.1007/s11042-023-15738-7

A deep learning framework for early diagnosis of Alzheimer's disease on MRI images

Doaa Ahmed Arafa[1] · Hossam El-Din Moustafa[2] · Hesham A. Ali[1,3] · Amr M. T. Ali-Eldin[1] · Sabry F. Saraya[1]

Received: 25 July 2022 / Revised: 14 March 2023 / Accepted: 25 April 2023 /
Published online: 19 May 2023

Abstract
Numerous medical studies have shown that Alzheimer's disease (AD) was present decades before the clinical diagnosis of dementia. As a result of the development of these studies with the discovery of many ideal biomarkers of symptoms of Alzheimer's disease, it became clear that early diagnosis requires a high-performance computational tool to handle such large amounts of data, as early diagnosis of Alzheimer's disease provides us with a healthy opportunity to benefit from treatment. The main objective of this paper is to establish a complete framework that is based on deep learning approaches and convolutional neural networks (CNN). Four stages of AD, such as (I) preprocessing and data preparation, (II) data augmentation, (III) cross-validation, and (IV) classification and feature extraction based on deep learning for medical image classification, are implemented. In these stages, two methods are implemented. The first method uses a simple CNN architecture. In the second method, the VGG16 model is the pre-trained model that is trained on the ImageNet dataset but applies the same model to the different datasets. We apply transfer learning, meaning, and fine-tuning to take advantage of the pre-trained models. Seven performance metrics are used to evaluate and compare the two methods. Compared to the most recent effort, the proposed method is proficient of analyzing AD, moreover, entails less labeled training samples and minimal domain prior knowledge. A significant performance gain on classification of all diagnosis groups was achieved in our experiments. The experimental findings demonstrate that the suggested designs are appropriate for basic structures with minimal computational complexity, overfitting, memory consumption, and temporal regulation. Besides, they achieve a promising accuracy, 99.95% and 99.99% for the proposed CNN model in the classification of the AD stage. The VGG16 pre-trained model is fine-tuned and achieved an accuracy of 97.44% for AD stage classifications.

Keywords Alzheimer's Disease (AD) · Convolution Neural Network (CNN) · Deep Learning (DL) · Transfer Learning (TL) · Imaging Pre-processing

✉ Doaa Ahmed Arafa
doaaarafa@mans.edu.eg

Extended author information available on the last page of the article

1 Introduction

Alzheimer's disease (AD) is the result of the degeneration of healthy brain cells and results in a continuous decline in memory, mental, and intellectual abilities. It is the most common cause of dementia, which affects mental and social skills. This impairs daily functioning in normal life and deteriorates further over time [22]. This phenomenon appears because of the death of nerve cells, the formation of amyloid plaques and neurofibrillary tangles, and the atrophy of the tissue throughout the brain, which becomes worse progressively [22, 35]. According to statistics from the World Health Organization, in 2021, around 55 million people worldwide will suffer from dementia, with the number expected to climb to 78 million in 2030 and 139 million in 2050, more than two times the number of dementia patients in 2021 [12].

Most people over 65 years of age have a high risk of getting dementia, whereas only 3% of young people have young-onset dementia that may be caused by different diseases [39]. Late treatment leads to damage to brain cells that are connected with the ability to think and memorize, which causes loss of brain function, a decrease in mental skills, language problems, and a decrease in the ability to construct logical thoughts. Starting with the gradual deterioration of nerve cells, the condition advances to an acute stage of dementia that leaves patients unable to carry out fundamental daily activities [36]. So, diagnosing AD in its early stages is very important because it progresses over time. AD is diagnosed through two methods: (1) when the patient has symptoms, and (2) by using neuroimaging techniques. AD shows up before any symptoms appear. This stage is referred to as "preclinical AD," during which no symptoms are noticeable by either the affected individual or those in their vicinity [13]. This stage of Alzheimer's disease can last for years, if not decades. Modern imaging technologies are capable of identifying amyloid beta deposits, a protein that is a hallmark of AD, regardless of whether any symptoms are perceptible [5]. The capacity to detect these early deposits could be particularly useful in clinical trials and in the future if novel AD treatments are developed.

Imaging techniques are the most prevalent approach for detecting AD because they allow for a non-invasive internal examination of the body. The history of AD treatment formerly faced a difficult period when the disease could only be discovered after death. But these days, medical imaging tools now play a significant role in the diagnosis and treatment of AD [33]. There are multiple neuroimaging techniques that have an essential role in the diagnosis of brain abnormalities, such as magnetic resonance imaging (MRI), positron emission tomography (PET), functional magnetic resonance imaging (fMRI), and computed tomography (CT). Although the most useful dataset for diagnosing AD is images, there are also handwritten and drawing task data sets that can be used in the diagnosis of patients with AD [11]. In our previous work [7], we showed the difference between them in detail. MRI and PET scanning are the most common imaging modalities that researchers use to diagnose AD. In this study, we combined MRI modalities that show brain structure and functionality with non-invasive techniques that help physicians and researchers detect Alzheimer's disease [7].

Nowadays, researchers are interested in machine learning in different fields, which has the ability to learn and improve algorithms and predict the solution to any problem. Where [11], proposed diagnosis AD based on machine learning techniques. They apply multi classifier architecture on handwritten and drawing task data like random forest, logistic regression, K-Nearest Neighbor, support vector machine, Gaussian Naive Bayes and Linear Discriminant Analysis. Deep learning (DL) is a subset of machine learning that has to support the best classification in many fields with the best performance, such as computer vision, diagnosis, and natural language processing [28].

Deep neural networks can identify small and complicated changes in brain structure using data, analyze the progression of AD, and provide reliable outcomes for the diagnosis of the disease [41]. DL has more than a few different types, such as convolutional neural networks (CNN), autoencoders, recurrent neural networks, and deep belief networks. CNN is the most well-known of the numerous deep learning models that are specifically built to deal with two-dimensional image data, although they can also be utilised with one-dimensional and three-dimensional data. The deep neural network needs a large dataset to train the model with the best performance [17]. In this paper, a methodology for diagnosing AD is proposed to achieve an accurate diagnosis and save the lives of many people. The proposed model works on the diagnosis of the MRI image to classify AD. This methodology aims to achieve high performance. Based on the issues raised above, we proposed a deep learning (DL) model to diagnose AD.

The objective of this paper is to present an end-to-end framework based on convolutional neural networks (CNN) that encompasses detailed steps, beginning with image acquisition and culminating in AD classification. The proposed machine learning application, supported by digital image processing, classifies scanned MRI images and predicts the presence and degree of Alzheimer's disease. The proposed framework achieves the following:

The following is a summary of the current paper's contributions:

- To balance our data collection, methods for data expansion have been implemented.
- To guarantee the unity of the given image, an image processing technique involving the transformation of images to a specific size is introduced.
- To Equilibrium the used dataset, data augmentation techniques have been implemented.
- During the categorization step, two models of the diagnosis are presented using the first method offered. The first proposed method used CNN architectures built from scratch. While the second method, the VGG16 model, is a pre-trained model that was trained on the ImageNet dataset but is applied to different datasets using transfer learning, meaning, and fine-tuning.
- To validate the proposed two approaches, a comparison and evaluation have been discussed and introduced based on the seven important performance metrics.

The rest of this paper is organized as follows: Section 2 presents different related works in this field. Section 3 shows the proposed algorithm and methodology. Experiments and results are shown in Section 4. Section 5 shows a discussion. Finally, this paper is concluded in Section 6.

2 Related work

Many studies in the realm of Alzheimer's disease diagnosis have recently been published. Several studies based on various datasets used various tools and approaches to aid in the categorization of AD. Table 1 shows the most recent methodologies in the classification of AD, datasets that are used, image types, number of images, and the performance of the techniques. According to recent research on the diagnosis of AD, there are two approaches used in training and classifying the model: traditional techniques and deep neural networks. [15, 21, 25] used traditional techniques to classify Alzheimer's disease, whereas [1-3, 22, 41] used deep neural networks techniques.

Table 1 Summarization of the previous DNN models

Paper	Dataset	Image type	Techniques	No. of images	Classification	Accuracy
AbdulAzeem et al. [1]	ADNI	MRI	CNN	211,655 after augmentation	AD vs CN	95.60%
Al-Khuzaie et al. [3]	OASIS	MRI	CNN (AlzNet model)	15,200	AD vs CN	97.88%
Al-Adhaileh [2]	Alzheimer's Dataset	MRI	AlexNet, ResNet50	1279	AD vs MCI vs NC	94.53%, 58.07%
Deepa et al. [10]	ADNI	MRI	VGG16 + AOA	819	CN vs AD vs MCI	92.34%
Zeng et al. [41]	ADNI	MRI	DBN	361	sMCI vs pMCI	87.78%
					HC vs AD	98.62%
					HC vs sMCI	92.31%
					HC vs pMCI	96.67%
					sMCI vs AD	99.62%
					pMCI vs AD	91.89%
Kong et al. [22]	ADNI	MRI + PET	3D-CNN	370	AD vs NC	93.21%
					MCI vs NC	86.52%
					AD vs MCI	85.63%
					AD vs MCI vs NC	87.67%
Meng et al. [25]	ADNI	fMRI	RS-SVM	1426	AD vs NC	91%
					LMCI vs NC	85%
					NC vs EMCI	86%
El-Sappagh et al. [15]	ADNI	MRI	LSTM	1371	AD vs MCI vs NC	93.87%
Kamal et al. [21]	OASIS	MRI	LS-SVM-RBF	416	AD vs CN	98%
			SVM			94%
			KNN			93%
			Random forest			68%
Antony et al. [6]	ADNI	MRI	VGG16	780	AD vs CN	81%
			VGG19			84%

Table 1 (continued)

Paper	Dataset	Image type	Techniques	No. of images	Classification	Accuracy
Liu et al. [23]	OASIS	MRI	CNN from scratch	–	AD vs CN	78.02%
			AlexNet			91.4%
			GoogLeNet			93.02%
Savaş et al. [32]	ADNI	MRI	EfficientNetB0	2182	AD vs MCI vs NC	92.98%
			EfficientNetB2			94.42%,
			EfficientNetB3			97.28%

AbdulAzeem et al. [1] proposed a CNN model that consists of three convolution layers and applies a max-pooling layer after every convolution layer for binary classification of AD. The Adam optimizer is used in the optimization process and uses the SoftMax activation function at the classify layer. They apply three experiments to show the effectiveness of the augmentation techniques on training datasets with different image sizes. Images are resized into two-scales (128, 128) and (64, 64). With both sizes, the model is trained with and without dropout to see how the dropout affects accuracy. The result shows the best accuracy when the image size is (128, 128) and without applying dropout. Cross-validation is done by splitting data into this range from 0.1 to 0.5. As the batch size increases, the training accuracy increases, but if the batch size increases to more than 64, the accuracy decreases. The accuracy of their proposed model achieves 95.6% accuracy for binary classification.

Al-Khuzaie et al. [3] proposed a CNN model named "Alzheimer Network" (AlzNet) for the binary classification of AD and CN. The AlzNet model is made up of five convolution layers, each with a ReLU activation function, and a max-pooling layer with a kernel size of 2 * 2. Their model was trained and tested on OASIS datasets that contained 15,200 images with a size of 200 × 200. The Adadelta optimizer was used to optimize the model with a ratio of 0.2 dropouts to overcome the overfitting problem. The model was trained with different numbers of dense units in the hidden layer to achieve the best performance, which was achieved at 121 units. The model achieves 97.99% training accuracy and 99.53% testing accuracy.

Al-Adhaileh et al. [2] suggested two pre-trained CNN models (AlexNet and Resnet50) to determine the best model for the diagnosis of AD. Their model was trained and tested on the Kaggle Alzheimer's dataset that was divided into 4 classes: MildDemented, ModerateDemented, NonDemented, and VeryMildDemented. The size of the input image is 224 × 224. The first proposed model is AlexNet, which has 34 layers and 5 max-pooling with size 4 × 4. They split the data into a 20% testing set and an 80% training set. The second model is ResNet50, which consists of 177 layers, 5 max-pooling with a size of 5 × 3, and an RMSprop optimizer. In both models, they used the ReLU activation function and the SoftMax function in the last layer to classify four classes. With 94.53% and 58.07% accuracy, respectively, the AlexNet model outperforms the ResNet50 model.

Antony et al. [6] suggested two models, VGG16 and VGG19, for the diagnosis of AD. Their model was trained on the 780-image ADNI dataset. The skull was stripped and augmented at the preprocessing stage before training the model. The input image size is 224×224. In VGG16 models, they used the sigmoid activation function in the last layer to classify two classes. In VGG19 models, they used the SoftMax activation function in the last layer to classify two classes. Where VGG16 has 64 and 128 kernel sizes, and VGG19 has 64, 128, and 256 kernel sizes. The accuracy in both models was insufficient; VGG-16 achieved 81% and VGG-19 achieved 84%.

Liu et al. [23] proposed a novel model for the classification of AD. Initially, a CNN model was developed from scratch, consisting of three convolutional layers, three pooling layers, and two fully connected layers, with SoftMax as the activation function in the last layer. Additionally, AlexNet and GoogLeNet models were also utilized, but with transfer learning applied to overcome overfitting issues and improve classification accuracy. However, the models did not yield significantly high classification accuracy. It should be noted that during transfer learning, both the AlexNet and GoogleNet models utilize 5-fold cross-validation and 500 training iterations. The classification accuracy rates achieved by the CNN model, AlexNet model, and GoogleNet model are 78.02%, 91.4%, and 93.02%, respectively. Since GoogleNet has more convolutions and deeper layers than AlexNet, it achieves a higher classification accuracy rate.

Savaş et al. [32] used different CNN models to classify the 2182 image objects that were taken from the ADNI database. The study gave a detailed framework for comparing the

performance of 29 models that had already been trained on the images. Preprocessing was applied to images by first converting the image format, cleaning the data, and splitting the images. After applying preprocessing techniques, image input is given to the models. During the test stage, the EfficientNetB0 model achieved the highest accuracy rate of 92.98%. In the comparative evaluation stage, the confusion matrix indicated that the EfficientNetB2 and EfficientNetB3 models achieved the highest precision, sensitivity, and specificity values for the Alzheimer's disease class, respectively, with rates of 94.42% and 97.28%. The study's results demonstrated that EfficientNet models performed remarkably well among the pre-trained models and achieved the highest classification performance.

3 The proposed methodology

As shown in Section 2, different studies have come up with different ways to diagnose AD. However, some studies didn't get the best results, while others got good results but had to use a more complicated and time-consuming model. So, in this paper, we try to improve the model and performance of the process of making a diagnosis while using as little computing power and memory as possible. The objectives of this article are to: (1) diagnose AD with the best performance possible; (2) enhance the accuracy of our proposed previous model in [14] by updating the model; (3) build a new CNN model; (4) compare two proposed models in training and testing with different performance metrics; and (5) discuss the effectiveness of the different optimizers on the same model and compare them according to diagnosis performance. In this study, the proposed schema of AD classification, as shown in Fig. 1, consists of four stages: the first stage is data preparation (data acquisition and preprocessing images), the second stage is data augmentation, the third stage is cross-validation, and the fourth stage is classification and feature extraction.

3.1 Dataset for the study

The dataset evaluated in this research is from the Kaggle Alzheimer's classification dataset [4]. The dataset has scans for testing and training, which consist of 1279 and 5121 scans, respectively. This data is divided into 717 MildDemented, 2560 NonDemented, 52 ModerateDemented, and 1792 VeryMildDemented. The dimension of the image is 176×208 and the format is jpg. In this study, we distinguish between two classes: mild-demented and non-demented. Dataset available on: https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images.

3.2 Data preparation

3.2.1 Data acquisition

The acquisition of input images is the initial step required for any medical image processing system. Images characterize the inner parts of the human body that can be captured using various imaging modalities with different procedures, as shown in Fig. 2. Neuroimaging techniques include (1) MRI, (2) PET, (3) fMRI, and (4) CT [34]. The choice of these modalities is based on the researcher's decision and the target task Datasets can be obtained from a variety of sources, including hospitals, clinics, radiology facilities, and online platforms [40].

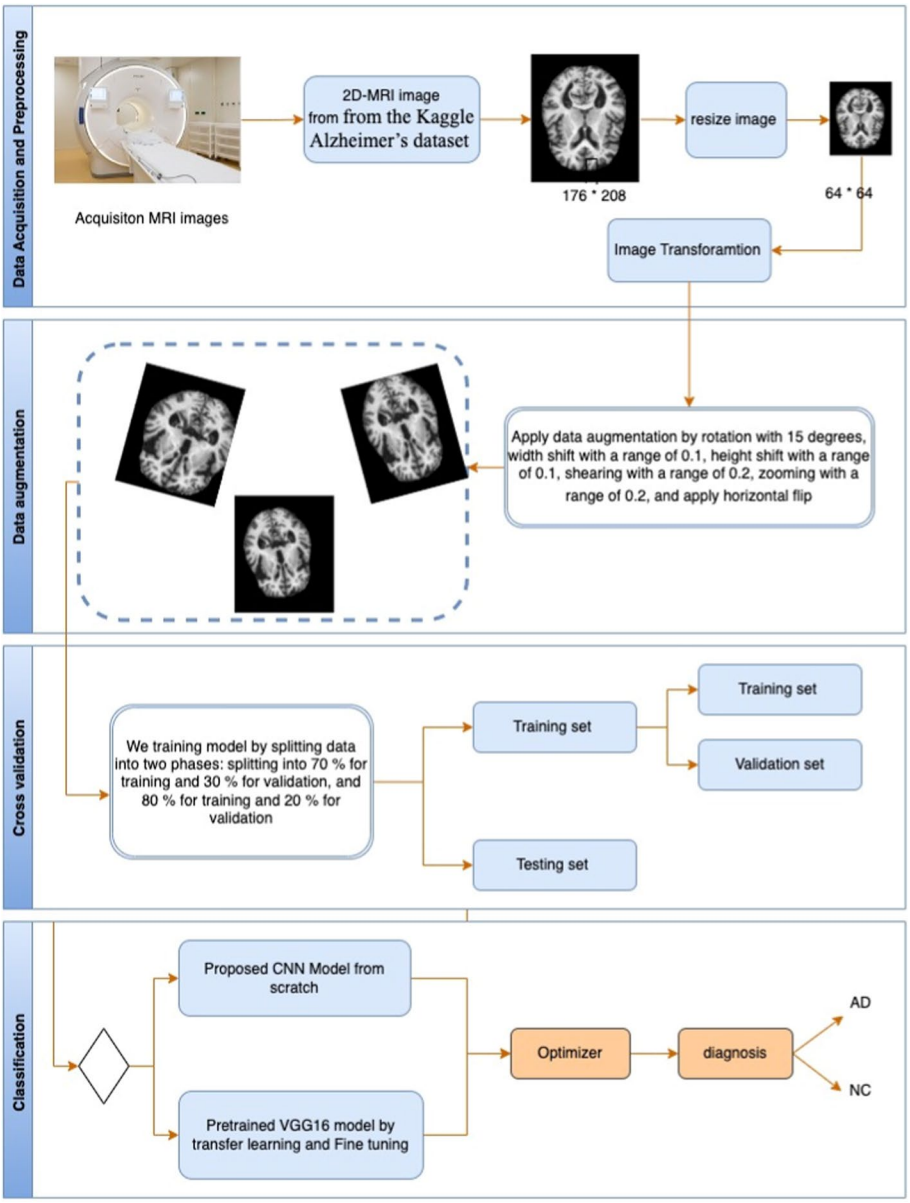


Fig. 1 Proposed Framework

3.2.2 Image preprocessing

Images are preprocessed in two steps: first, the image is transformed, and then it is normalized. The transformation process involves resizing the input image from 176×208 to 64×64. Resizing the image is a critical process in model training, as the model trains faster when the image size is smaller. After resizing the image, normalization is applied to

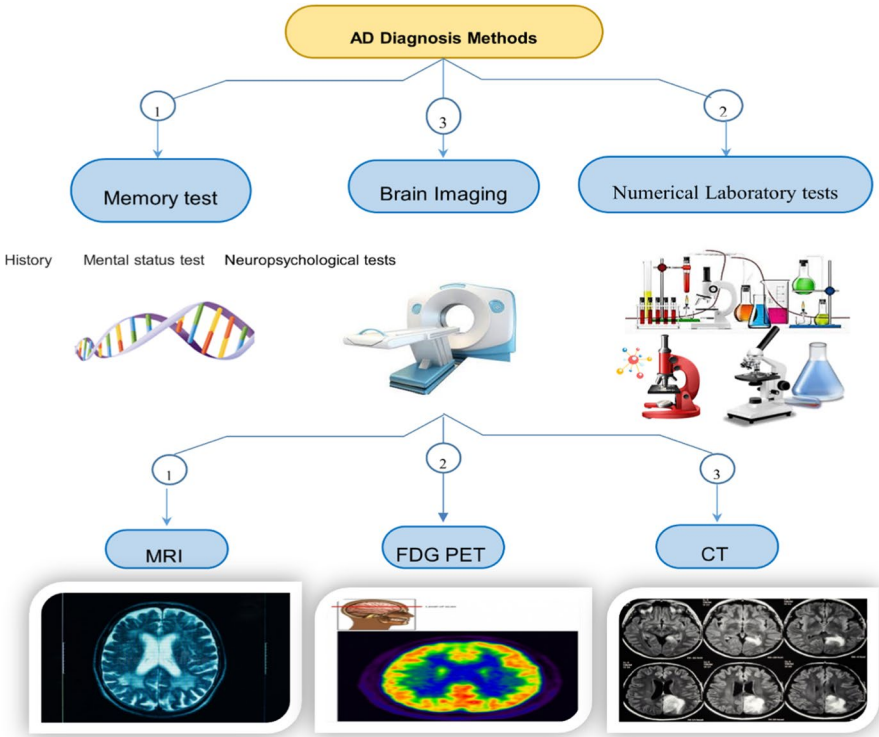


Fig. 2 Different Imaging Techniques for AD Diagnosis [7]

each of them. We normalize the image. Normalization is the process of scaling each pixel in the image from a specific range (0 to 255) to a value between 0 and 1 [18]. After doing both the transformation and normalization, the new images are stored in a pickle file.

3.2.3 Image augmentation

The data augmentation process improves the imbalance problem. In the proposed model, to overcome the possible over-fitting problem in the training CNN models and to combine possible image discrepancies, augmented images were generated from the original slices by six operations: rotation, translation, gamma correction, random noise addition, scaling, and random affine transformation. At this stage, the dataset size is increased by using the augmentation technique. By sufficiently shifting the brain's position, it is possible to achieve this goal and stop a model from memorizing the brain's location. The classic MRI protocol includes the following operations:

- Rotation: rotation of an image without cropping because the cropped image may not contain the whole tumor; images have been rotated at 15 angles.
- Mirroring: are right- or left-mirrored.
- Flipping: Images are up-down flipped.

We apply augmentation techniques with the following settings: rotation at 15 degrees, width shift with a range of 0.1, height shift with a range of 0.1, shearing with a range of 0.2, zooming with a range of 0.2, and applying horizontal flip. We balanced the dataset to increase the accuracy of classification by augmenting images and copying some of the images randomly to balance classes. After augmenting the dataset, MildDemented has 2519 images, and NonDemented has 2561 images. The augmented data were added to enhance the original training dataset to allow for a sufficiently large sample size. As shown in Fig. 3, this method increases the training and validation groups of MR images for axial.

3.3 Cross-Validation

Cross-validation is a common technique used to evaluate the performance of deep learning models. It involves dividing the available dataset into two or more subsets: one for training the model and the other(s) for evaluating its performance. The most common type of cross-validation used in deep learning is k-fold cross-validation [29]. In k-fold cross-validation, the dataset is divided into k subsets, or "folds" of approximately equal size. The model is then trained k times, each time using a different fold as the validation set and the remaining folds as the training set [8]. The results from each of the K-fold

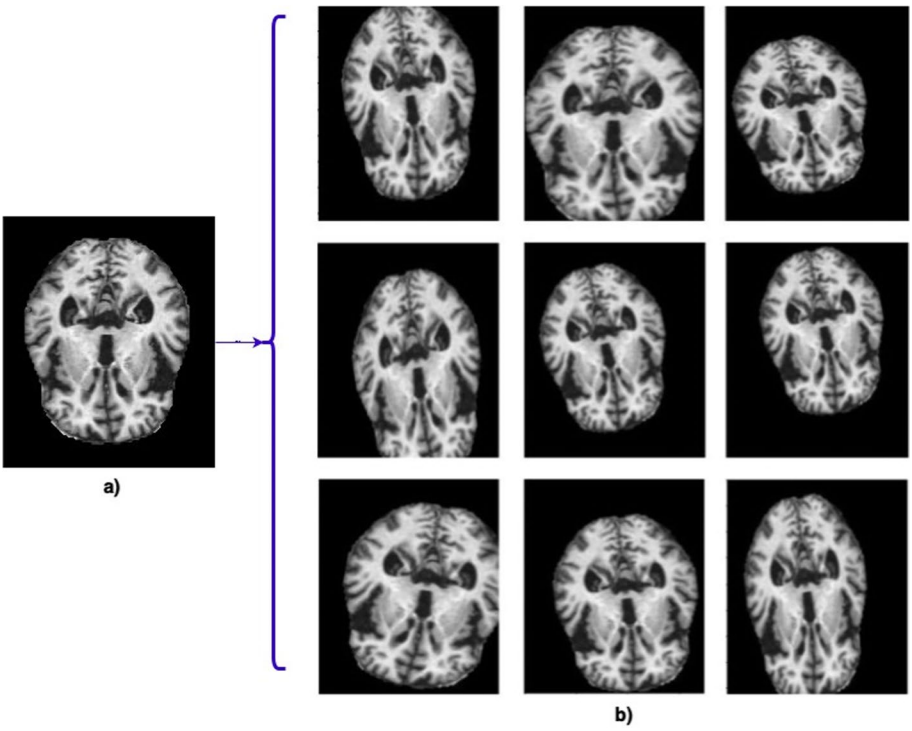


Fig. 3 Samples for applying augmentation techniques. Where (**a**) is original image that has disease, (**b**) after applying augmentation techniques

models are then averaged to obtain a final estimate of the model's performance. Cross-validation is necessary in deep learning for several reasons [37]:

- Performance evaluation: Cross-validation provides a more accurate estimate of the model's performance on unseen data than simply evaluating the model on a single holdout validation set. By averaging the performance across multiple validation sets, cross-validation reduces the variance of the performance metric and provides a more representative estimate of the model's performance.
- Hyperparameter tuning: Deep learning models have many hyperparameters that need to be tuned to achieve optimal performance. Cross-validation allows us to systematically test different hyperparameter configurations and select the one that produces the best performance on unseen data.
- Overfitting prevention: Deep learning models are prone to overfitting, which occurs when the model performs well on the training data but poorly on the validation or test data. Cross-validation can help detect overfitting by evaluating the model's performance on multiple validation sets and averaging the results.
- Data efficiency: In some cases, the dataset may be limited in size. Cross-validation allows us to maximize the use of the available data by using all the available data for training and validation instead of reserving a large portion for testing.

Our dataset is composed of two main parts: the first folder contains training data, and the second contains testing data. We perform cross-validation in many forms to detect the best performance and classification. First, we split the training dataset into 80% training and 20% validation sets. Second, we split the training dataset into 70% training and 30% validation sets. The main objective of this process is to train the model with a different ratio of the training and testing sets and achieve the best performance for the diagnosis of AD.

3.4 Feature extraction and classification by deep learning

Neural network techniques have recently been popular in medical image classification, computer vision, healthcare, and speech recognition, with the best performance. It is a form of artificial neural network that has neurons that transmit the signal to another neuron, whose output is determined by a non-linear function of the sum of its inputs [24]. Neural network techniques are based on certain complex algorithms that can extract high-level characteristics from data and use those features to classify and solve problems. DL has various algorithms such as CNN, Autoencoder (AE), Deep Belief Network (DBN), Restricted Boltzmann Network (RBN), Deep Neural Network (DNN), Recurrent Neural Network (RNN), etc. [7, 9].

CNN considers this the most popular deep neural network architecture. One of the most essential features of CNN is its ability to handle enormous datasets and achieve good classification performance. CNN has several pre-trained models that are trained on the ImageNet dataset [19], such as VGG16, LeNet, GoogLeNet, ResNet, and AlexNet [31]. In this paper, we used the CNN technique, which is used mainly in object detection and image processing. CNN consists of several layers; each layer has a specific task to help the model extract the feature, such as the convolution layer, max-pooling layer, and fully connected layer. We will discuss it in detail in the next section.

3.4.1 Feature extraction

Feature extraction is one of the most important concepts in deep learning, especially for computer vision. In deep learning, a neural network learns to extract relevant features or patterns from raw input data, such as images, audio, or text, in order to perform a specific task, such as classification, regression, or generation [38]. The purpose of feature extraction is to transform the raw data into a set of features that can be used as inputs to a machine learning model, which then learns to classify or predict new data based on these features [16]. Feature extraction is typically performed using pre-trained deep neural networks, such as convolutional neural networks (CNNs) for images. These networks are trained on large datasets to learn high-level representations of the raw data, which can then be used to extract features from new data.

Deep learning models can learn to automatically extract relevant features from raw data, but feature extraction can still be useful in some cases, particularly when the dataset is small, or the features are known to be important [20]. Feature extraction: Feature extraction involves transforming the input data into a set of features. This can be done using techniques such as principal component analysis (PCA), wavelet analysis, or convolutional neural networks (CNNs), depending on the nature of the data.

3.4.2 Convolution layer

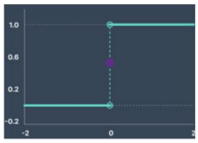
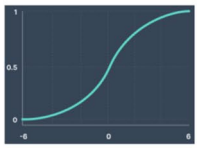
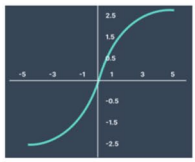
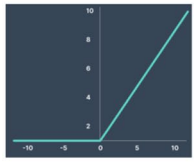
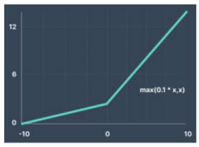
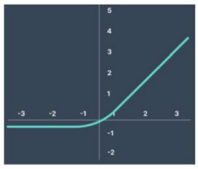
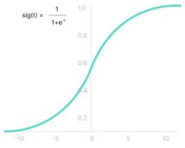
The main part of CNN is the convolution layer, which extracts feature maps by bypassing the kernel at a specific size. This kernel is applied to the input image by sliding the filter across the image and multiplying element by element, to produce a different feature and passing it to the next layer. In the neural network, there are several hidden layers, each of which extracts a specific feature. The first layer usually extracts the basic features, like the edges of the images. Then, the output feature is input for the next layer, which extracts more complex features such as corners. By going to deep layers, it extracts the more complex features, such as objects and faces.

The activation function is used to help the neural network learn complex problems and determine which neuron should be activated or not. The main purpose of the activation function is to transform the input into a non-linear output node to allow it to learn complex tasks. This activation function is added to all hidden layers, and they all have the same activation function. However, the output layer uses a different activation function depending on the classification type. We have used several types of activation functions, which we summarize in Table 2.

3.4.3 Pooling layer

After extracting features with the convolution layer, we have a large number of parameters that require more computation power. To reduce the computational power required to process the data, pooling layers are used to reduce the dimensions of feature maps. As a result, the number of parameters to learn and the amount of processing in the network are both reduced. Pooling layers have two types: max pooling and average pooling. Max pooling is done by sliding a filter over each feature map that is extracted by the convolution layer and selecting the maximum value above this filter. Average pooling is the same concept as max pooling, but it calculates the average of the values above the filter. Thus, the output is a feature map calculated as in Eq. (1) and Eq. (2).

Table 2 Summarization of the activation function, where x refers to input of activation function

Activation function	Equation	Curve
Binary step	$f(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases}$	
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	
Tanh	$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	
ReLU	$f(x) = \begin{cases} x \; for \; x > 0 \\ 0 \; for \; x \leq 0 \end{cases}$	
Leaky ReLU	$f(x) = \begin{cases} x \; for \; x > 0 \\ 0.1x \; for \; x \leq 0 \end{cases}$	
ELU	$f(x) = \begin{cases} x & for \; x \geq 0 \\ \alpha(e^{x} - 1) & for \; x < 0 \end{cases}$	
Softmax	$f(x) = \frac{e^{x_i}}{\sum_{j=1} e^{x_j}}$	

$$Output\ of\ Width = \frac{W - F + 2P}{S} + 1 \tag{1}$$

$$Output\ of\ Height = \frac{H - F + 2P}{S} + 1 \tag{2}$$

Where W is the width of the feature map, H is the height of the feature map, F is the size of the filter, P is the value of padding if not equal to zero, and S is the stride value.

3.4.4 Fully connected layer

Every neuron in the fully connected layer is linked to every neuron in the next layer. It converts a multi-dimensional feature map to a vector with one dimension. The final few layers in most typical DL models are fully connected layers that compile the data retrieved by the preceding layers to generate the final output. In the last layer, the number of neurons depends on the type of classification, which is either binary or multi-classification. For binary classification, the neuron in the last layer has two neurons, but for multiclassification, the number of neurons equals the number of classes.

3.5 The proposed architecture

We have proposed two architectures to compare them and choose the one that achieves the best performance. Each model has the same input images with the same size of (64×64). The first model is trained from scratch, whereas the second is a pre-trained VGG16 model. These models extract features and classify them automatically, not manually, as is done in machine learning. In the next subsection, a detailed description of each of them is provided.

3.5.1 The First Proposed CNN architecture

Figure 4 and Table 3 show the summary of the first proposed CNN model. Our model was trained on the preprocessed images with a size of 64×64. We are splitting our trained data into two cross-validations: (1) 20% testing and 80% training; and (2) 30% testing and 70% training. It consists of three convolution layers with a (3,3) kernel size, a ReLU activation function, and padding of type "same." After each convolution layer, we add a 2D max-pooling layer of different sizes (3,3) and (2,2) as shown in Table 3. Flatten is used to convert all of the feature map's 2D dimension arrays to a single vector. We used different dropout ratios of 0.2 and 0.5 to show the effectiveness of the dropout in the proposed model. The last layer is the fully connected layer with a sigmoid activation function to classify the data into two classes. We used different training parameters. The number of epochs was used as 10, 15, and 20, where the batch size was 16, 32, 64, and 128.

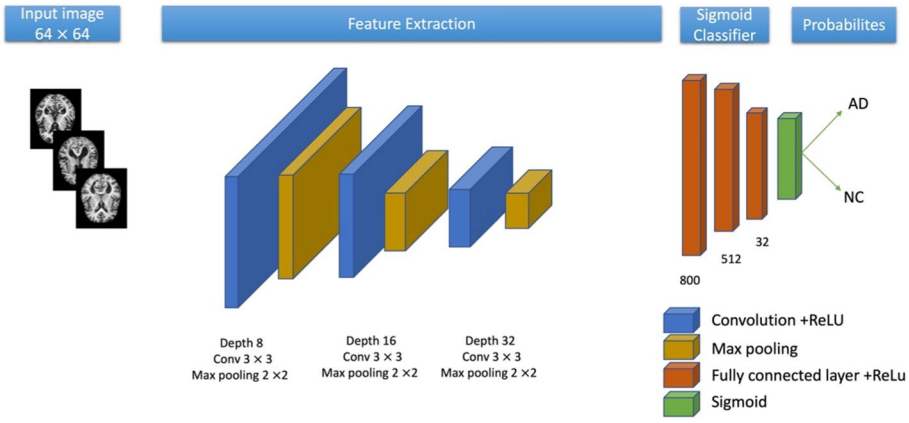


Fig. 4 First Proposed CNN model

Table 3 The architecture of the first proposed CNN

Layer	Activation function	Output shape	Pool size
2D convolution layer	ReLU	64 × 64 × 8	
2D Max pooling layer	–	21 × 21 × 8	(3, 3)
2D convolution layer	ReLU	21 × 21 × 16	
2D max-pooling layer	–	10 × 10 × 16	(2,2)
2D convolution layer	ReLU	10 × 10 × 32	
2D max-pooling layer	–	5 × 5 × 32	(2,2)
Flatten layer	–	800	
Dense layer	ReLU	512	
Dropout layer	–	512	
Dense layer	ReLU	32	
Dropout layer	–	32	
Dense layer	Sigmoid	2	

Algorithm 1 depicts the steps of our proposed CNN model so that you can understand how it works. The inputs to the model are datasets. Before training the proposed model, the input image must be preprocessed; the first process is the transformation, and the second resizes the image to a suitable size. The dataset is split into training and testing sets. We train our model at different values of epochs and batch sizes. Then we used Adam's optimizer to compile the model. The output of the proposed model is the performance metrics (loss, accuracy, precision, recall, specificity, and f-score) and graphs. Then we apply the same steps for the second experiment but split the data into different ratios for the training and testing sets.

3.5.2 The Second Pretrained VGG16 architecture

The VGG16 model is a pre-trained model that was trained on the ImageNet dataset but can be applied to other datasets as well. We use transfer learning and fine-tuning. In transfer learning, the model is trained on different small datasets, and the layer is frozen to avoid destroying the information. In this model, we are freezing 16 layers of convolution and pooling to keep the feature extraction the same and adding a new classifier layer to be compatible with our dataset. Due to the first layer's ability to extract simpler and more general features, we freeze the layer that extracts general features and train the layer that extracts complicated patterns depending on the dataset. So, we used fine-tuning, which is the approach of transfer learning. In fine-tuning, we do not freeze all convolution and pooling layers. The final convolution and pooling layer have been unfrozen. We retrained only the last layer of convolution and pooling and added three fully connected layers as shown in Table 4.

Figure 5 and Table 4 show the summary of the second proposed VGG16 model. Our model was trained on the preprocessed image with a size of 64 × 64. We are splitting our trained data into 40% testing and 60% training. We apply the augmentation process to images with a 15° rotation range, a 0.1 width and height shift, a 0.2 shear, and a 0.2 zoom factor. The first 16 layers are frozen and retrained at the last convolution and pooling layer. We are adding dropout with a ratio of 0.2 to avoid the overfitting problem. The number of epochs and batch size were used (50, 250, and 512) and (32, and 64), respectively. Different optimizers (Adam, SGD, Adadelta, RMSprop, and Adagrad) are applied to our model to show the effectiveness of each of them on the data.

Table 4 The architecture of the Second pre-trained VGG16 model

Layer	Activation function	Output shape	Pool size	Trainable
Input layer		64 ×64×3		False
2D convolution layer	ReLU	64 ×64×64		False
2D convolution layer	ReLU	64 ×64×64		False
2D Max pooling layer		32 ×32×64	(2, 2)	False
2D convolution layer	ReLU	32 ×32×128		False
2D convolution layer	ReLU	32 ×32×128		False
2D Max pooling layer		16 ×16×128	(2, 2)	False
2D convolution layer	ReLU	16 ×16×256		False
2D convolution layer	ReLU	16 ×16×256		False
2D convolution layer	ReLU	16 ×16×256		False
2D Max pooling layer		8 ×8×256	(2, 2)	False
2D convolution layer	ReLU	8 ×8×512		False
2D convolution layer	ReLU	8 ×8×512		False
2D convolution layer	ReLU	8 ×8×512		False
2D Max pooling layer		4 ×4×512	(2, 2)	False
2D convolution layer	ReLU	4 ×4×512		False
2D convolution layer	ReLU	4 ×4×512		False
2D convolution layer	ReLU	4 ×4×512		True
2D Max pooling layer		2 ×2×512	(2, 2)	True
2D global average pooling		512		True
Dense layer	ReLU	1024		True
Dropout layer	–	1024		True
Dense layer	ReLU	512		True
Dropout layer	–	512		True
Dense layer	Sigmoid	2		True

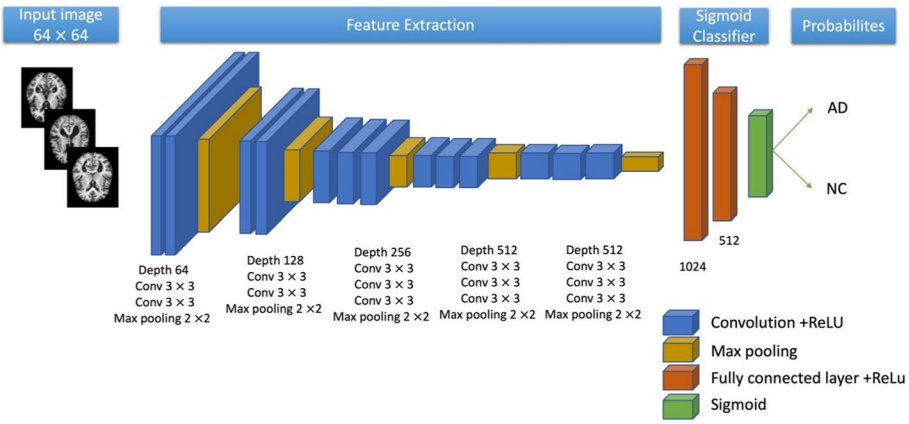


Fig. 5 Second VGG16 model

Algorithm 1 Proposed CNN Model

```
Input: Set of Image form the Alzheimer's Dataset, N: no of images
Output: Model, Result, Graphs
begin
// Image processing
For i = 1 to N
        Transformation by using fit_transform ();
        Resize image from 176×208 to 64 ×64;
Next i
Establishing and preparing the Learning model
Initialization
//First experiment
        Dived Data set 80% for training and 20% for testing
        Set the epochs to (10, 15, 20)
        Set the batch size to (16, 32, 64, and 128)
        For i = 1 to N*0.8
                Apply Adam Optimizer
                Compile the model and apply loss function (Binary cross entropy)
                Train the model and saving model
                Get Results and Graphs
        Next i
// Testing phase
        For i = 1 to N*0.2
                Loading model
                Evaluate model on tested data
                Measure confusion metric (loss, accuracy, precision, specificity, recall, f1-score)
                Get Results and Graphs
        Next i
// Second experiment
        Dived Data set 70% for training and 30% for testing
        For i = 1 to N*0.7
                Apply Adam Optimizer
                Compile the model and apply loss function (Binary cross entropy)
                Train the model and saving model
        Next I
End
```

Algorithm 2 demonstrates the steps of the pre-trained VGG16 model with fine tuning and transfer learning. The inputs to the model are MR images. Before training the proposed model, the input images must be preprocessed, which consists of two operations; the first is the transformation operation, and the second is resizing the image to a suitable size. The dataset is split into training and testing sets. We train our model at different values of

Algorithm 2 Proposed VGG16 Model

```
Input: Set of Image form the Alzheimer's Dataset
N: no of images
Output: Model, Result, Graphs
begin
// Image processing
For i = 1 to N
        Transformation by using fit_transform ();
        Resize image from 176×208 to 64 ×64;
Next i
Establishing and preparing the Learning model
Initialization
//First experiment
        Dived Data set 60% for training and 40% for testing
        Set the epochs to (50, 250, 512)
        Set the batch size to 32
        For i = 1 to N*0.6
                Implement VGG16 model and Apply Transfer Learning and Fine tuning
                Apply different Optimizer (Adam, SGD, Adagrad, Adadelta, RMSprop)
                Apply Augmentation techniques.
                Compile the model and apply loss function (Binary cross entropy)
                Train the model and saving model
                Get Results and Graphs
        Next i
// Testing phase
        For i = 1 to N*0.2
                Loading model
                Evaluate model on tested data
                Measure confusion metric (loss, accuracy, precision, specificity, recall, f1-score)
                Get Results and Graphs
        Next i
End
```

epochs and batch sizes. To get the best performance, we need to increase the datasets by using data augmentation techniques. The model compiles with different optimizers (Adam, Adagrad, Adadelta, SGD, and RMSprop). The output of the proposed model is the performance metrics (loss, accuracy, precision, recall, specificity, and f-score) and graphs. Then we tested the proposed model to evaluate the training model.

3.6 Performance evaluation parameters

Recent studies have used the confusion matrix to analyze a model and provide information about the classification performance since it is robust to categorize data relationships and any distribution. The confusion matrix gives valuable information for classification models based on different metrics [30]. First, four primary keys are used to test the classifier: true positive (Tp), true negative (Tn), false positive (Fp), and false negative (Fn) values. Then, based on the

four outcome values, the performance of the model will be calculated: accuracy (ACC), sensitivity (Recall), specificity (SPC), precision (PPV), F1-score.

Accuracy, as given in eq. (3), is the number of correctly predicted to the total predicted number.

$$\textbf{\textit{Accuracy}}\ (ACC) = \frac{\mathrm{Tp} + \mathrm{Tn}}{\mathrm{Tp} + \mathrm{Tn} + F\mathrm{p} + F\mathrm{n}} \tag{3}$$

Sensitivity (Recall), as given in eq. (4), is the number of samples predicted to be positive from the total number of samples that are positive, also known as the true positive rate.

$$\textbf{\textit{Sensitivity}}(Recall) = \frac{T\mathrm{p}}{T\mathrm{p} + F\mathrm{n}} \tag{4}$$

Specificity (SPC, TNR) for the true negative rate, as given in eq. (5) is the number of samples predicted as negative from the total number of samples negative.

$$\textbf{\textit{Specificity}}\ (SPC) = \frac{Tn}{Tn + Fp} \tag{5}$$

Precision, as in Eq. (6), also called positive predictive value, represents the number of samples actually and predicted as positive from the total number of samples predicted as positive.

$$\textbf{\textit{Precision}}\ (PPV) = \frac{Tp}{Tp + Fp} \tag{6}$$

The harmonic means of precision and recall, known as the F1-score, is shown in eq. (7).

$$\textbf{\textit{F1}} - \textbf{\textit{score}} = \frac{2 * Tp}{2 * Tp + Fp + F\mathrm{n}} \tag{7}$$

4 Experiments and results

In our study, the developed system's processing and classification used the Python programming language, and the models were trained using Google Colab and a graphical processing unit. For the purpose of formally determining whether the first or the second model has a significant improvement, we conducted three experiments with three configurations. The first and second are for figuring out how to classify AD when splitting data with different ratios. In experiment 1, we split the data into 20% testing and 80% training, whereas in experiment 2, we split the data into 30% testing and 70% training. And both of these experiments used different ratios of dropout (0.2 and 0.5) to demonstrate the efficacy of the dropout in the proposed model. The final layer is a fully connected layer that utilizes a sigmoid activation function to classify the data into two classes. We utilized various training parameters. The number of epochs was 10, 15, and 20, and the batch sizes were 16, 32, 64, and 128.

The third looked at how well the VGG16 model worked with different optimizers and transfer learning. The initial sixteen layers are preserved, while the final convolution and pooling layers are retrained. To avoid the overfitting issue, we are implementing a dropout ratio of 0.2. There were 50, 250, and 512 epochs, and the batch sizes were 32 and 64, respectively. Several optimizers (Adam, SGD, Adadelta, RMSprop, and Adagrad) are applied to our model in order to demonstrate the efficacy of each on the data.

4.1 Experiment 1

Table 5 shows the diagnostic performance metrics of experiment 1 on the training dataset (on the first proposed model) and the effect of the model with different batch sizes, number of epochs, and with or without dropout ratio. Batch size values are 16, 32, 64, and 128. The number of epoch values is 10, 15, and 20 with or without dropout. The best training

Table 5 Training result of experiment 1

No. of epochs	Dropout	Batch size	Loss	Accuracy	Precision	Recall	AUC	F1	specificity
10	without	16	0.0118	99.68	0.9968	0.9968	0.9999	0.9968	0.9924
		32	0.0081	99.85	0.9985	0.9985	1.0000	0.9985	0.9932
		64	0.0203	99.68	0.9968	0.9966	1.0000	0.9967	0.9824
		128	0.1420	96.06	0.9593	0.9603	0.9912	0.9598	0.8938
	0.2	16	0.0363	98.72	0.9870	0.9877	0.9988	0.9873	0.9816
		32	0.0128	99.66	0.9966	0.9968	0.9999	0.9967	0.9908
		64	0.0291	99.29	0.9922	0.9927	0.9997	0.9924	0.9782
		128	0.0619	98.42	0.9846	0.9847	0.9983	0.9847	0.9523
	0.5	16	0.0279	99.04	0.9904	0.9904	0.9991	0.9904	0.9831
		32	0.0166	99.53	0.9953	0.9956	0.9999	0.9954	0.9882
		64	0.0375	99.04	0.9893	0.9905	0.9992	0.9899	0.9722
		128	0.1033	97.14	0.9719	0.9687	0.9940	0.9703	0.9286
15	without	16	6.5023e-05	100	1.0000	1.0000	1.0000	1.0000	0.9999
		32	2.4942e-04	100	1.0000	1.0000	1.0000	1.0000	0.9997
		64	0.0043	100	1.0000	1.0000	1.0000	1.0000	0.9958
		128	0.0035	100	1.0000	1.0000	1.0000	1.0000	0.9966
	0.2	16	1.8109e-04	100	1.0000	1.0000	1.0000	1.0000	0.9998
		32	0.0069	99.78	0.9978	0.9978	0.9997	0.9978	0.9956
		64	0.0098	99.78	0.9971	0.9980	1.0000	0.9976	0.9922
		128	0.0182	99.68	0.9971	0.9963	0.9999	0.9967	0.9854
	0.5	16	0.0304	98.97	0.9890	0.9894	0.9994	0.9892	0.9834
		32	0.0394	98.67	0.9870	0.9870	0.9988	0.9870	0.9771
		64	0.0204	99.36	0.9937	0.9934	0.9998	0.9935	0.9861
		128	0.0397	98.82	0.9902	0.9871	0.9992	0.9886	0.9715
20	Without	16	4.0351e-05	100	1.0000	1.0000	1.0000	1.0000	1.0000
		32	1.1094e-04	100	1.0000	1.0000	1.0000	1.0000	0.9999
		64	2.4002e-04	100	1.0000	1.0000	1.0000	1.0000	0.9998
		128	0.0096	99.95	0.9995	0.9995	1.0000	0.9995	0.9910
	0.2	16	6.1363e-05	100	1.0000	1.0000	1.0000	1.0000	0.9999
		32	2.2789e-04	100	1.0000	1.0000	1.0000	1.0000	0.9998
		64	8.7885e-04	100	1.0000	1.0000	1.0000	1.0000	0.9992
		128	0.0039	100	1.0000	1.0000	1.0000	1.0000	0.9960
	0.5	16	0.0014	99.93	0.9993	0.9993	1.0000	0.9993	0.9989
		32	0.0146	99.51	0.9951	0.9951	0.9996	0.9951	0.9915
		64	0.0040	99.9	0.9990	0.9995	1.0000	0.9993	0.9967
		128	0.0051	99.95	0.9995	0.9995	1.0000	0.9995	0.9950

accuracy (100%) is achieved without dropout and with an increasing number of epochs and decreasing batch size. The best average accuracy of training results, 99.95%, is achieved at 32 batch sizes and without applying dropout. To further demonstrate the effectiveness of the proposed framework, the training performance of the model is evaluated with different batch sizes and dropout ratios as shown in Fig. 6, each for the first and the second experiment. Based on the results presented in this figure, we can observe that the best accuracy achieved at first experiment with 32-batch size and without using dropout.

Table 6 shows the performance metric of experiment 1 with the testing dataset (on the first proposed model). By comparing Table 5, the best testing accuracy is achieved at batch sizes of 32 without applying dropout, and the number of epochs is 20. The best average accuracy of the testing result was achieved at 16 batch size and without applying dropout, with 0.2 ratios and 32 batch size, and with 0.5 ratios and 32 batch size, with 99.98, 99.84, and 99.81, respectively.

4.2 Experiment 2

Table 7 shows the performance metric of experiment 2 with the training dataset (on the first proposed model) and the effect of the model with different batch sizes, number of epochs, and with or without dropout ratio. The data was split into a 30% testing set and a 70% training set. Batch size values are 16, 32, 64, and 128. The number of epoch values is 10, 15, and 20 with or without dropout. The best training accuracy (100%) is achieved without dropout and with an increasing number of epochs and decreasing batch size. The best average accuracy of training results, 99.99%, is achieved at a 32-batch size and without applying dropout.

Table 8 shows the performance metric of experiment 2 with the testing dataset (on the first proposed model). By comparing Table 7. The best testing accuracy is achieved at batch size 32, with a 0.2 dropout ratio, and the number of epochs is 20. The best average accuracy of the testing result was achieved at 32 batch size and without applying dropout, with 0.2 ratios and 16 batch size, and with 0.5 ratios and 16 batch size, at 99.79, 99.9, and 99.53, respectively.

4.3 Experiment 3

Table 9 shows the performance metric of experiment 3 with the training dataset (on the second VGG16 proposed model) and the effect of the model with different batch sizes,

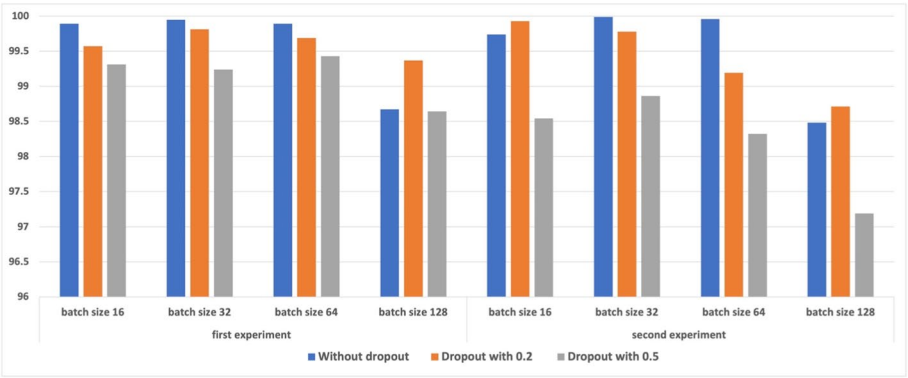


Fig. 6 Comparison between first and second experiment, for each experiment we perform training with different batch size and with different dropout ratio

Table 6 Testing result of experiment 1 (Where bold text represents the best accuracy)

No. of epochs	Dropout	Batch size	Loss	Accuracy	Precision	Recall	AUC	F1	specificity
10	without	16	0.002	99.96	0.99960	0.996658	0.99999	0.99960	0.99821
		32	0.00443	99.921	0.99921	0.99921	0.99978	0.99921	0.99698
		64	0.01178	99.94	0.99941	0.99941	0.9999	0.99941	0.98930
		128	0.11347	96.673	0.96691	0.966428	0.99486	0.96666	0.91443
	0.2	16	0.0321	99.074	0.99057	0.99076	0.99918	0.99066	0.97875
		32	0.00684	99.803	0.99803	0.99803	0.99977	0.99803	0.996538
		64	0.00859	99.822	0.99824	0.99843	0.99998	0.99834	0.99237
		128	0.04067	99.3108	0.99336	0.99374	0.99944	0.99355	0.96658
	0.5	16	0.04222	98.641	0.98643	0.98643	0.99858	0.98643	0.97511
		32	0.00491	99.94	0.99941	0.99941	0.99999	0.99941	0.99598
		64	0.04721	97.834	0.97648	0.98016	0.99864	0.97830	0.96743
		128	0.06608	97.814	0.9776	0.97881	0.99783	0.97824	0.95548
15	without	16	0.00026	100	1.0	1.0	0.9999	1.0	0.99976
		32	0.00216	99.94	0.99941	0.99941	0.99999	0.99941	0.99878
		64	0.0060	99.862	0.99863	0.99863	0.99999	0.99863	0.99509
		128	0.0067	99.803	0.99804	0.99804	0.99994	0.99804	0.99519
	0.2	16	0.00515	99.901	0.99901	0.99901	0.999602	0.99901	0.998842
		32	0.0065	99.763	0.99764	0.99764	0.9999	0.99764	0.99705
		64	0.01098	99.66	0.99633	0.99633	0.9999	0.99633	0.99144
		128	0.01202	99.803	998,040.	0.99804	0.9999	0.99804	0.99101
	0.5	16	0.00697	99.901	0.99901	0.99901	0.99994	0.99901	0.99607
		32	0.0035	99.901	0.99901	0.99901	0.99979	0.99901	0.99800
		64	0.01863	99.31	0.9931	0.99257	0.99977	0.9928	0.99066
		128	0.0897	99	0.96841	0.96387	0.99534	0.96613	0.94322
20	**Without**	**16**	**0.00041**	**100**	**1.0**	**1.0**	**1.0**	**1.0**	**0.99963**
		32	**0.00048**	**100**	**1.0**	**1.0**	**1.0**	**1.0**	**0.99954**
		64	**0.0006**	**100**	**1.0**	**1.0**	**1.0**	**1.0**	**0.99939**
		128	0.0107	99.842	0.99843	0.99824	0.99974	0.99833	0.99211
	0.2	16	0.0008	99.98	0.9998	0.9998	0.99999	0.9998	0.99953
		32	0.00190	99.961	0.99961	0.99961	0.9998	0.99961	0.99941
		64	0.00573	99.881	0.99882	0.99882	0.999598	0.998828	0.99844
		128	0.00420	99.881	0.998828	0.998828	0.9999	0.99882	0.99704
	0.5	16	0.01413	99.724	0.99724	0.99724	0.998795	0.99724	0.99675
		32	0.00993	99.606	0.99626	0.99587	0.99993	0.99606	0.99477
		64	0.0006	99.98	0.99980	0.99980	1.0	0.99980	0.99946
		128	0.00280	99.921	0.99922	0.99922	0.99999	0.99922	0.99833

number of epochs, and different optimizers such as Adam, SGD, Adadelta, RMSprop, and Adagrad. The data was split into a 40% testing set and a 60% training set. Batch size values are 32 and 64. The number of epoch values is 50, 250, and 512. Experiment 3 shows the Adam optimizer achieves the best accuracy of other optimizers by 97.44%. This result was achieved after 512 epochs and with 64 batch sizes. The RMSprop has a higher accuracy of 94.42% than the other. To further demonstrate the effectiveness of the proposed framework, the training performance of the model is evaluated with different batch sizes and

Table 7 Performance metric for experiment 2

No. of epochs	Dropout	Batch size	Loss	Accuracy	Precision	Recall	AUC	F1	specificity
10	without	16	0.0219	99.21	0.9922	0.9922	0.9997	0.9922	0.9863
		32	0.0034	99.97	0.9997	0.9997	1.0000	0.9997	0.9968
		64	0.0177	99.89	0.9989	0.9989	1.0000	0.9989	0.9837
		128	0.0741	97.95	0.9798	0.9790	0.9977	0.9794	0.9444
	0.2	16	0.0077	99.80	0.9980	0.9975	1.0000	0.9977	0.9948
		32	0.0214	99.47	0.9950	0.9941	0.9995	0.9946	0.9854
		64	0.0578	97.83	0.9789	0.9788	0.9983	0.9788	0.9612
		128	0.1244	96.43	0.9630	0.9627	0.9909	0.9629	0.9154
	0.5	16	0.0630	97.86	0.9791	0.9793	0.9964	0.9791	0.9672
		32	0.0783	97.10	0.9722	0.9710	0.9962	0.9715	0.9514
		64	0.1163	96.12	0.9610	0.9604	0.9913	0.9607	0.9263
		128	0.1929	93.81	0.9363	0.9369	0.9781	0.9366	0.8660
15	without	16	9.9447e-05	100	1.0000	1.0000	1.0000	1.0000	0.9999
		32	4.1816e-04	100	1.0000	1.0000	1.0000	1.0000	0.9996
		64	0.0019	100	1.0000	1.0000	1.0000	1.0000	0.9982
		128	0.0811	97.5	0.9749	0.9751	0.9972	0.9750	0.9384
	0.2	16	2.4695e-04	100	1.0000	1.0000	1.0000	1.0000	0.9997
		32	0.0052	99.89	0.9986	0.9989	1.0000	0.9987	0.9963
		64	0.0115	99.75	0.9969	0.9975	1.0000	0.9972	0.9905
		128	0.0134	99.83	0.9980	0.9983	1.0000	0.9981	0.9883
	0.5	16	0.0416	98.59	0.9861	0.9871	0.9980	0.9866	0.9778
		32	0.0132	99.58	0.9953	0.9955	0.9999	0.9954	0.9912
		64	0.0150	99.77	0.9972	0.9978	0.9999	0.9975	0.9885
		128	0.0484	98.79	0.9886	0.9874	0.9987	0.9880	0.9654
20	Without	16	3.2688e-05	100	1.0000	1.0000	1.0000	1.0000	0.9999
		32	1.8533e-04	100	1.0000	1.0000	1.0000	1.0000	0.9998
		64	2.1659e-04	100	1.0000	1.0000	1.0000	1.0000	0.9998
		128	0.0080	100	1.0000	1.0000	1.0000	1.0000	0.9925
	0.2	16	4.8936e-05	100	1.0000	1.0000	1.0000	1.0000	0.9999
		32	5.5469e-04	100	1.0000	1.0000	1.0000	1.0000	0.9995
		64	0.0019	100	1.0000	1.0000	1.0000	1.0000	0.9981
		128	0.0092	99.86	0.9986	0.9986	1.0000	0.9986	0.9920
	0.5	16	0.0269	99.18	0.9916	0.9916	0.9989	0.9916	0.9871
		32	0.0037	99.89	0.9989	0.9989	1.0000	0.9989	0.9970
		64	0.0363	99.07	0.9903	0.9894	0.9987	0.9898	0.9773
		128	0.0351	98.99	0.9911	0.9908	0.9993	0.9909	0.9753

different optimizers, as shown in Fig. 7. Based on the results presented in this figure, we can observe that Adam optimizers achieved the best accuracy with a 64-batch size.

Table 10 shows the performance metric of experiment 3 with the testing dataset (on the first proposed model). By comparing Table 9. The Adam optimizer achieved the best testing accuracy of 93.25% when the batch size was 64 and after 512 epochs. Also, RMSprop has the best value among other optimizers, achieving 92.87%.

Table 8 Testing result of experiment 2 (Where bold text represents the best accuracy)

No. of epochs	Dropout	Batch size	Loss	Accuracy	Precision	Recall	AUC	F1	specificity
10	without	16	0.01862	99.389	0.99391	0.99410	0.99942	0.99401	0.99089
		32	0.006394	99.822	0.9982	0.99823	0.99998	0.99823	0.99535
		64	0.02514	99.744	0.99746	0.99746	0.99994	0.99746	0.97767
		128	0.073679	97.479	0.9744	0.97425	0.99645	0.97434	0.95224
	0.2	16	0.00721	99.842	0.99842	0.99843	0.99976	0.998427	0.99659
		32	0.01128	99.606	0.99606	0.99587	0.99993	0.99596	0.99342
		64	0.01605	99.763	0.99731	0.99730	0.99991	0.99730	0.986364
		128	0.09005	97.932	0.97932	0.97989	0.99605	0.97961	0.93144
	0.5	16	0.01867	99.369	0.99345	0.99345	0.99959	0.99345	0.98899
		32	0.03706	98.995	0.98989	0.98989	0.99945	0.98989	0.97305
		64	0.08910	97.066	0.96951	0.97137	0.99626	0.97043	0.93550
		128	0.13591	95.806	0.95785	0.95804	0.99161	0.9579	0.89567
15	without	16	0.00476	99.803	0.99803	0.99803	0.99979	0.99803	0.99774
		32	0.00452	99.803	0.99803	0.99803	0.99998	0.99803	0.99691
		64	0.00895	99.704	0.99707	0.99707	0.99976	0.99707	0.99461
		128	0.06450	98.444	0.98457	0.9839	0.99793	0.98428	0.95107
	0.2	16	0.00295	99.901	0.99901	0.99901	0.99979	0.99901	0.99859
		32	0.00307	99.921	0.99921	0.99901	0.99999	0.99911	0.99799
		64	0.02519	99.193	0.99161	0.9922	0.99913	0.99190	0.986543
		128	0.00895	99.881	0.99882	0.99882	0.99998	0.99882	0.99241
	0.5	16	0.01720	99.468	0.99425	0.9944	0.99927	0.99434	0.99277
		32	0.01640	99.389	0.99371	0.9939	0.99929	0.99381	0.99300
		64	0.00667	99.881	0.9988	0.99882	0.99995	0.9988	0.99542
		128	0.023775	99.488	0.99492	0.99492	0.99979	0.99492	0.98196
20	Without	16	0.00276	99.82	0.99823	0.99823	0.99999	0.99823	0.99841
		32	0.00649	99.76	0.99764	0.99764	0.99978	0.99764	0.99730
		64	0.00427	99.763	0.99785	0.99765	0.99999	0.99775	0.99743
		128	0.01281	99.803	0.99824	0.99804	0.99995	0.99814	0.98888
	0.2	**16**	**0.00061**	**99.98**	**0.99980**	**0.99980**	**0.999999**	**0.99980**	**0.99948**
		32	0.00526	99.76	0.99764	0.99764	0.99998	0.99764	0.99738
		64	0.010065	99.66	0.99649	0.9970	0.99937	0.99677	0.99562
		128	0.006545	99.881	0.99882	0.99882	0.99999	0.99882	0.99424
	0.5	16	0.00888	99.763	0.99765	0.99783	0.99995	0.9977	0.996701
		32	0.00375	99.842	0.99842	0.99842	0.99999	0.99842	0.998350
		64	0.01257	99.566	0.995703	0.995703	0.99991	0.99570	0.99221
		128	0.02144	99.704	0.99707	0.99707	0.9999	0.99707	0.98168

Table 11 shows a comparison between the proposed models and the other state-of-the-art techniques. The result showed the proposed model (an experimental one) achieved the best result compared with others. In Experiment 1, training was proposed for CNN from scratch at a 16-batch size without dropout. Moreover, Fig. 8 shows a comparison of our developed models with other previous models. Based on the results presented in this figure, we can observe that our model achieved the best performance.

Table 9 Training result for experiment 3 (Where bold text represents the best accuracy)

Optimizer	No. epochs	Batch size	Loss	Accuracy	AUC	Precision	Recall	F1
Adam	50	32	0.3359	84.25	0.9291	0.8422	0.8436	0.8429
		64	0.3186	85.17	0.9363	0.8528	0.8522	0.8525
	250	32	0.1324	94.65	0.9887	0.9472	0.9466	0.9469
		64	0.1275	95.21	0.9897	0.9518	0.9523	0.9520
	512	32	0.1008	95.96	0.9934	0.9600	0.9593	0.9596
		64	**0.0673**	**97.44**	**0.9968**	**0.9746**	**0.9749**	**0.9748**
SGD	50	32	0.4134	80.54	0.8903	0.8046	0.8061	0.8053
		64	0.4455	78.8	0.8710	0.7885	0.7898	0.7891
	250	32	0.2480	88.97	0.9623	0.8882	0.8904	0.8893
		64	0.3014	87.13	0.9438	0.8719	0.8727	0.8723
	512	32	0.1458	94.16	0.9869	0.9412	0.9402	0.9407
		64	0.1756	92.78	0.9813	0.9283	0.9271	0.9277
Adadelta	50	32	0.5921	71.12	0.7703	0.7034	0.7104	0.7061
		64	0.6210	68.82	0.7355	0.6702	0.6745	0.6719
	250	32	0.4745	77.85	0.8520	0.7777	0.7793	0.7782
		64	0.4889	76.76	0.8434	0.7684	0.7660	0.7670
	512	32	0.4281	80.21	0.8823	0.8018	0.7991	0.8002
		64	0.4559	79.03	0.8655	0.7899	0.7893	0.7894
RMSprop	50	32	0.3766	83.98	0.8398	0.8403	0.8411	0.8407
		64	0.3579	83.98	0.9199	0.8397	0.8405	0.8401
	250	32	0.2511	90.88	0.9661	0.9074	0.9083	0.9078
		64	0.2311	90.74	0.9681	0.9074	0.9086	0.9080
	512	32	0.2144	92.35	0.9733	0.9235	0.9220	0.9228
		64	**0.1659**	**94.42**	**0.9843**	**0.9435**	**0.9438**	**0.9437**
Adagrad	50	32	0.4522	79.19	0.8662	0.7908	0.7884	0.7894
		64	0.4752	78.04	0.8523	0.7798	0.7762	0.7779
	250	32	0.3312	85.72	0.9327	0.8572	0.8518	0.8543
		64	0.3956	81.92	0.9007	0.8182	0.8208	0.8194
	512	32	0.2238	90.55	0.9705	0.9069	0.9069	0.9068
		64	0.3034	87.82	0.9440	0.8780	0.8783	0.878

5 Discussion

With a high death rate, AD is considered one of the most common types of irreversible dementia worldwide, eventually causing death. Early detection of AD at early stages could enhance the survival opportunities of the patients and lead to better drug effects. In this study, we extensively examined the role of applying deep learning models with different architectures in AD diagnosis. The study contributes to a new end-to-end DL-based model for differentiating between AD and NC. Our proposed model has several stages for the diagnosis of AD: the first is for preparing the dataset by processing it to minimize the size of the image to reduce the complexity of the computational process; the second is to apply augmentation techniques to overcome the overfitting problem; the third is to apply cross validation; and finally, we apply our proposed model that will be trained on processed data with different optimizers to optimize the diagnosis of AD.

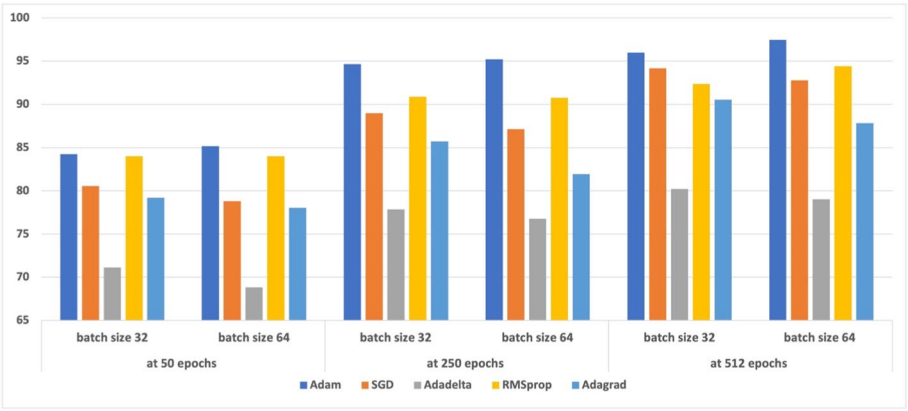


Fig. 7 Comparison for VGG16 model with different optimizers (Adam, SGD, Adadelta, RMSprop, Adagrad) and at different epochs and batch size

By comparing other studies with our proposed model. Liu et al. [6], built a CNN model, and the accuracy reached 78.02%. However, they proposed two pretrained models, AlexNet and GoogleNet, that achieved 91.4% and 93.02% respectively. Liu et al. [6] note that the limitation of this study is that it does not achieve high accuracy, and the number of images or cases not mentioned in the article. Al-Khuzaie et al. [1] proposed the CNN model. Although it has more images than other studies, it did not achieve high accuracy compared to our proposed model. Whereas in the proposal by Antony et al. [2], the accuracy reached 81%. Although Murugan et al. [27] and Mggdadi et al. [26] used the same dataset (the Kaggle Alzheimer's dataset), we used it in our proposed model. They didn't achieve a proper performance. Mggdadi et al. [26] suggested two models, CNN and VGG16, that attained 67.5% and 70.3% accuracy, respectively. And Murugan et al. [27] they proposed CNN model which achieved 95.23%.

Based on these results, no work achieved high performance on Kaggle Alzheimer's dataset, we believe that our proposed VGG16 model and CNN model has better performance than other methods, verifying the effectiveness of CNN for AD diagnosis. Whereas our proposed CNN model reaches 99.98% accuracy and the accuracy of our VGG16 model reaches 97.44%. Our proposed architecture provided promising results. However, there are number of limitations that need to be addressed in order to go forward with further clinical trials. Firstly, the number of cases need to be increased. Secondly, the dataset is acquired from one hospital. Thirdly, binary classification where there are multi classes for AD. One of the future works will be to investigate integrating the proposed framework into clinical workflow as a decision support tool and diagnosis multiclass of AD.

6 Conclusion

This study proposed a framework for the diagnosis of AD that focused on deep learning and CNN architecture. Two models are proposed. First, the CNN model was trained from scratch with different ratios of dropout and by splitting the data into different training and testing set

Table 10 Testing result for experiment 3 (Where bold text represents the best accuracy)

Optimizer	No. epochs	Batch size	Loss	Accuracy	AUC	Precision	Recall	F1
Adam	50	32	0.3346	85.0954	0.929819	0.85124	0.85126	0.85124
		64	0.304726	86.8478	0.94250	0.86891	0.86668	0.86779
	250	32	0.253124	90.7068	0.9679	0.90741	0.907395	0.90739
		64	0.194611	92.912	0.980285	0.92935	0.929534	0.929445
	512	32	0.20051	92.715	0.97883	0.927203	0.927203	0.927203
		64	**0.194786**	**93.2467**	**0.981633**	**0.932650**	**0.932464**	**0.93255**
SGD	50	32	0.48296	75.979	0.848482	0.75906	0.759715	0.759376
		64	0.484968	76.373	0.85054	0.76443	0.76442	0.764426
	250	32	0.259417	89.151	0.96002	0.8914	0.89182	0.891584
		64	0.29935	86.709	0.94592	0.86740	0.869420	0.86841
	512	32	0.208413	91.986	0.97666	0.920764	0.920204	0.920474
		64	0.29935	86.709	0.94592	0.86740	0.86942	0.86840
Adadelta	50	32	0.56585	75.94	0.815319	0.74744	0.780592	0.780592
		64	0.59864	75.0344	0.795573	0.74420	0.76158	0.75268
	250	32	0.47929	78.0862	0.84983	0.78038	0.780036	0.78018
		64	0.475055	78.8934	0.85221	0.789596	0.784502	0.787012
	512	32	0.46321	78.617	0.86334	0.78667	0.78664	0.78664
		64	0.467264	78.735	0.858135	0.78923	0.78645	0.78782
RMSprop	50	32	0.35539	83.697	0.92138	0.8363	0.83691	0.83659
		64	0.34385	84.15	0.92557	0.84251	0.84149	0.84199
	250	32	0.38105	88.718	0.94948	0.88737	0.887578	0.88746
		64	0.397293	85.134	0.93812	0.85195	0.85145	0.85170
	512	32	0.274246	92.577	0.969265	0.92564	0.92582	0.925733
		64	**0.21084**	**92.8726**	**0.979273**	**0.92947**	**0.92877**	**0.929121**
Adagrad	50	32	0.45857	78.617	0.86309	0.79000	0.781019	0.785418
		64	0.46184	79.503	0.86064	0.795362	0.79876	0.79704
	250	32	0.366825	83.7763	0.917521	0.84028	0.83620	0.838197
		64	0.40866	81.098	0.894409	0.81091	0.81516	0.81302
	512	32	0.40866	81.098	0.8944	0.810913	0.81516	0.81302
		64	0.31476	86.6903	0.93937	0.866242	0.86825	0.867233

Table 11 Comparison the proposed technique with other state-of-the-art techniques (Where bold text represents the best accuracy of our proposed model)

Research	Model	Accuracy
Al-Khuzaie et al. [3]	CNN (AlzNet model)	97.88%
Al-Adhaileh [2]	AlexNet, ResNet50	94.53%, 58.07%
Deepa et al. [10]	VGG16 + AOA	92.34%
Antony et al. [6]	VGG16	81%
Mggdadi et al. [26]	VGG16, CNN	70.3%, 67.5%
Proposed Experimental 1	**CNN (80% training set)**	**99.98%**
Proposed Experimental 2	**CNN (70% training set)**	**99.9%**
Proposed Experimental 3	**VGG16 + transfer learning**	**97.44%**

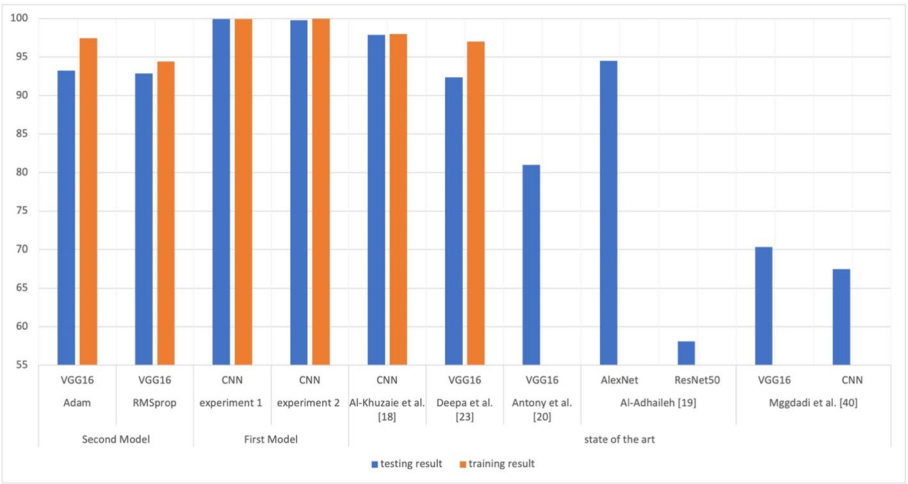


Fig. 8 Comparison between all experiments with the best testing and training accuracy

Table 12 Table of Abbreviations

Abbreviation	Definition
AD	Alzheimer's Disease
ADNI	Alzheimer's Disease Neuroimaging Initiative
AE	Autoencoder
AOA	Arithmetic Optimization Algorithm
CNN	Convolution Neural network
CT	Computed Tomography
DBN	Deep Belief Network
DNN	Deep Neural Network
ELU	Exponential Linear Units
EMCI	early mild cognitive impairment
fMRI	functional Magnetic Resonance Imaging
HC	Health Control
LMCI	late mild cognitive impairment
MRI	Magnetic Resonance Imaging
MRI	Magnetic Resonance Imaging
NC	Normal Controls
OASIS	Outcome and Assessment Information Set
PET	Positron Emission Tomography
RBN	Restricted Boltzmann Network
RNN	Recurrent Neural Network
RS-SVM	Random Survey Support Vector Machines
SGD	Stochastic gradient descent
VGG16	Visual Geometry Group

ratios. Second, we used a pre-trained VGG16 model, applying transfer learning and fine-tuning and measuring the effectiveness of different optimizers (Adam, Adagrad, Adadelta, SGD, RMSprop) with the model. The evaluation and comparison of the two methods implement seven performance metrics. The experimental results indicate that the proposed architectures are appropriate for simple structures with low computational complexity, memory consumption, overfitting, and controllable time. The proposed models achieved very promising accuracies by comparing with other researchers, as we present in the result section: 99.95% and 99.99% for the proposed CNN model in the classification of the AD stage. The VGG16 pre-trained model is fine-tuned and has achieved an accuracy of 97.44% for AD stage classifications. In the future, we intend to apply hyperparameter optimization techniques to each model and perform multi-classification for AD stages.

Table of Abbreviations Table 12 presents the used abbreviations in the current survey with the corresponding definitions. They are sorted in alphabetical order.

Acknowledgments The authors would like to express appreciation to Prof Hesham Arafat, who assisted in the research work, and to Mansoura University for supporting the authors with the dataset collection.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB). The authors declare that they did not receive any funding for the current study.

Data availability We confirm that no datasets were users, generated, or analyzed during the current study.

Declarations

Authors agreement We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm thatthe order of authors listed in the manuscript has been approved by all of us.

Corresponding author We understand that the declared corresponding author is the sole contact for the editorial process (including Editorial Manager and direct communications with the office). He/she is responsible forcommunicating with the other authors about progress, submissions of revisions, and final approval of proofs. We confirm that we have provided a current, correct email address that is accessible by theCorresponding Author.

Intellectual property We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, concerning intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

Conflict of interest The authors declare that they have no competing interests nor conflict of interests for the current study.

References

1. AbdulAzeem Y, Bahgat WM, Badawy M (2021) A CNN based framework for classification of Alzheimer's disease. Neural Comput Applic 33(16):10415–10428. https://doi.org/10.1007/s00521-021-05799-w
2. Al-Adhaileh MH (2022) Diagnosis and classification of Alzheimer's disease by using a convolution neural network algorithm. Soft Comput 26:7751–7762. https://doi.org/10.1007/s00500-022-06762-0
3. Al-Khuzaie FEK, Bayat O, Duru AD (2021) Diagnosis of Alzheimer Disease Using 2D MRI Slices by Convolutional Neural Network. Appl Bionics Biomech 2021:1–9. https://doi.org/10.1155/2021/6690539
4. "Alzheimer's Dataset (4 class of Images) | Kaggle." (n.d.) https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images. Accessed 20 May 2022
5. "Alzheimer's stages: How the disease progresses - Mayo Clinic." (n.d.) https://www.mayoclinic.org/diseases-conditions/alzheimers-disease/in-depth/alzheimers-stages/art-20048448. Accessed 07 Apr 2022
6. Antony F, Anita HB, George JA (2023) *Classification on Alzheimer's Disease MRI Images with VGG-16 and VGG-19*, vol. 312. https://doi.org/10.1007/978-981-19-3575-6_22
7. Arafa DA, Moustafa HE-D, Ali-Eldin AMT, Ali HA (2022) Early detection of Alzheimer's disease based on the state-of-the-art deep learning approach: a comprehensive survey. Multimed Tools Appl 81:23735–23776. https://doi.org/10.1007/s11042-022-11925-0
8. Berrar D (n.d.) Cross-Validation Call for Papers for Machine Learning journal: Machine Learning for Soccer View project Cross-validation. https://doi.org/10.1016/B978-0-12-809633-8.20349-X
9. Bhangale KB, Kothandaraman M (2022) Survey of Deep Learning Paradigms for Speech Processing. Wirel Pers Commun 125:1913–1949. https://doi.org/10.1007/s11277-022-09640-y
10. Deepa N, Chokkalingam SP (2022) Optimization of VGG16 utilizing the Arithmetic Optimization Algorithm for early detection of Alzheimer's disease. Biomed Signal Process Control 74:103455. https://doi.org/10.1016/J.BSPC.2021.103455
11. De Gregorio G, Desiato D, Marcelli A, Polese G (2021) *A Multi Classifier Approach for Supporting Alzheimer's Diagnosis Based on Handwriting Analysis*, vol. 12661 LNCS. https://doi.org/10.1007/978-3-030-68763-2_43
12. "Dementia." (n.d.) https://www.who.int/news-room/fact-sheets/detail/dementia. Accessed 07 Apr 2022
13. Dubois B et al (2016) Preclinical Alzheimer's disease: Definition, natural history, and diagnostic criteria. Alzheimer's Demen 12(3):292–323. https://doi.org/10.1016/j.jalz.2016.02.002
14. Ebrahim D, Ali-Eldin AMT, Moustafa HE, Arafat H (2020) Alzheimer Disease Early Detection Using Convolutional Neural Networks. In: *Proceedings of ICCES 2020–2020 15th International Conference on Computer Engineering and Systems*. https://doi.org/10.1109/ICCES51560.2020.9334594
15. El-Sappagh S, Saleh H, Ali F, Amer E, Abuhmed T (2022) Two-stage deep learning model for Alzheimer's disease detection and prediction of the mild cognitive impairment time. Neural Comput Applic 34:14487–14509. https://doi.org/10.1007/s00521-022-07263-9
16. "Feature Extraction Definition | DeepAI." (n.d.) https://deepai.org/machine-learning-glossary-and-terms/feature-extraction. Accessed 27 Feb 2023
17. "How Do Convolutional Layers Work in Deep Learning Neural Networks?" (n.d.) https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/. Accessed 12 May 2022
18. "How to Manually Scale Image Pixel Data for Deep Learning." (n.d.) https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/. Accessed 20 May 2022
19. "ImageNet." (n.d.) https://www.image-net.org/index.php. Accessed 24 May 2022
20. Ishaque M, Hudec L (May 2019) Feature extraction using Deep Learning for Intrusion Detection System. *2nd International Conference on Computer Applications and Information Security, ICCAIS 2019*. https://doi.org/10.1109/CAIS.2019.8769473
21. Kamal M et al (2022) Machine Learning and Image Processing Enabled Evolutionary Framework for Brain MRI Analysis for Alzheimer's Disease Detection. Comput Intell Neurosci 2022:1–8. https://doi.org/10.1155/2022/5261942
22. Kong Z, Zhang M, Zhu W, Yi Y, Wang T, Zhang B (2022) Multi-modal data Alzheimer's disease detection based on 3D convolution. Biomed Signal Process Control 75:103565. https://doi.org/10.1016/J.BSPC.2022.103565
23. Liu J, Li M, Luo Y, Yang S, Li W, Bi Y (2021) Alzheimer's disease detection using depthwise separable convolutional neural networks. Comput Methods Prog Biomed 203:106032. https://doi.org/10.1016/J.CMPB.2021.106032

24. Mehmood A, Abugabah A, AlZubi AA, Sanzogni L (2022) Early Diagnosis of Alzheimer's Disease Based on Convolutional Neural Networks. Comput Syst Sci Eng 43(1):305–315. https://doi.org/10.32604/csse.2022.018520
25. Meng X, Wu Y, Liu W, Wang Y, Xu Z, Jiao Z (2022) Research on Voxel-Based Features Detection and Analysis of Alzheimer's Disease Using Random Survey Support Vector Machine. Front Neuroinform 16:856295. https://doi.org/10.3389/fninf.2022.856295
26. Mggdadi E, Al-Aiad A, Al-Ayyad MS, Darabseh A (May 2021) Prediction Alzheimer's disease from MRI images using deep learning. *2021 12th International Conference on Information and Communication Systems, ICICS 2021*, pp. 120–125. https://doi.org/10.1109/ICICS52457.2021.9464543
27. Murugan S et al (2021) DEMNET: A Deep Learning Model for Early Diagnosis of Alzheimer Diseases and Dementia from MR Images. IEEE Access 9:90319–90329. https://doi.org/10.1109/ACCESS.2021.3090474
28. Poloni KM, Ferrari RJ (2022) A deep ensemble hippocampal CNN model for brain age estimation applied to Alzheimer's diagnosis. Expert Syst Appl 195:116622. https://doi.org/10.1016/j.eswa.2022.116622
29. Raschka S (2018) Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning
30. Ruuska S, Hämäläinen W, Kajava S, Mughal M, Matilainen P, Mononen J (2018) Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. Behav Process 148:56–62
31. Sarraf S, Tofighi G, Org S (2016) Classification of Alzheimer's Disease Structural MRI Data by Deep Learning Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1607.06583
32. Savaş S (2022) Detecting the Stages of Alzheimer's Disease with Pre-trained Deep Learning Architectures. Arab J Sci Eng 47(2):2201–2218. https://doi.org/10.1007/s13369-021-06131-3
33. Shanmugam JV, Duraisamy B, Simon C, Bhaskaran P (2022) Alzheimer's disease classification using pre-trained deep networks. Biomed Signal Process Control 71:103217. https://doi.org/10.1016/j.bspc.2021.103217
34. Theodore WH, Dorwart R, Holmes M, Porter RJ, DiChiro G (1986) Neuroimaging in refractory partial seizures: Comparison of PET, CT, and MRI. Neurology 36(6):750–759. https://doi.org/10.1212/wnl.36.6.750
35. Tufail AB, Ma Y-K, Zhang Q-N (2020) Binary Classification of Alzheimer's Disease Using sMRI Imaging Modality and Deep Learning. J Digit Imaging 33(5):1073–1090. https://doi.org/10.1007/s10278-019-00265-5
36. Turkson RE, Qu H, Mawuli CB, Eghan MJ (2021) Classification of Alzheimer's Disease Using Deep Convolutional Spiking Neural Network. Neural Process Lett 53(4):2649–2663. https://doi.org/10.1007/s11063-021-10514-w
37. "What is Cross-Validation?. Testing your machine learning models… | by Mohammed Alhamid | Towards Data Science." (n.d.) https://towardsdatascience.com/what-is-cross-validation-60c01f9d9e75. Accessed 24 Feb 2023
38. "What is Feature Extraction? Feature Extraction in Image Processing | Great Learning." (n.d.) https://www.mygreatlearning.com/blog/feature-extraction-in-image-processing/. Accessed 27 Feb 2023
39. "World Alzheimer Report 2021 | Alzheimer's Disease International (ADI)." (n.d.) https://www.alzint.org/resource/world-alzheimer-report-2021/. Accessed 08 Apr 2022
40. Wu O et al (2019) Big data approaches to phenotyping acute ischemic stroke using automated lesion segmentation of multi-center magnetic resonance imaging data. Stroke 50(7):1734–1741. https://doi.org/10.1161/STROKEAHA.119.025373
41. Zeng N, Li H, Peng Y (2021) A new deep belief network-based multi-task learning for diagnosis of Alzheimer's disease. Neural Comput Applic. https://doi.org/10.1007/s00521-021-06149-6

Doaa Ahmed Arafa Teaching Assistant at the Department of computer engineering and control systems, Faulty of Engineering, Mansoura University.

Hossam El-Din Moustafa Professor at the Department of Electronics and Communications Engineering, the founder and former executive manager of the Biomedical Engineering Program (BME) at the Faculty of Engineering, Mansoura University. He is an IEEE senior member. Research interests include biomedical imaging, image processing applications, and bioinformatics.

Hesham Arafat Ali He is currently a Dean in the Faculty of Artificial Intelligence, Delta University for Science and Technology, Gamasa, and a professor in the computer engineering and control systems department at the faculty of engineering, Mansoura University.

Amr A. T. Ali-Eldin Associate professor in computer engineering and control systems department at the faculty of engineering, Mansoura University.

Sabry Fouad Saraya Associate professor in computer engineering and control systems department at the faculty of engineering, Mansoura University.

Authors and Affiliations

Doaa Ahmed Arafa[1] · Hossam El-Din Moustafa[2] · Hesham A. Ali[1,3] · Amr M. T. Ali-Eldin[1] · Sabry F. Saraya[1]

[1] Computer Engineering and Control Systems Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt

[2] Electronics and Communication Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt

[3] Faculty Artificial Intelligence, Delta University for Science and Technology, Gamasa, Mansoura, Egypt