Multimedia Tools and Applications
https://doi.org/10.1007/s11042-022-13831-x

A recognition model for handwritten Persian/Arabic numbers based on optimized deep convolutional neural network

Saqib Ali[1] · Sana Sahiba[1] · Muhammad Azeem[2] · Zeeshan Shaukat[1] · Tariq Mahmood[3] · Zareen Sakhawat[1] · Muhammad Saqlain Aslam[4]

Received: 19 January 2021 / Revised: 1 March 2022 / Accepted: 6 September 2022

Abstract
Recently, artificial intelligence-based applications are universally acknowledged. Digit recognition, particularly Persian/Arabic handwritten digits, has many applications in today's commercial contexts for example office automation and document processing. However, researcher are struggling in hand-crafted digit scripts due to the presence of different digit writing patterns, cursive nature and lack of large public databases that make the feature extraction process more complex. Therefore, critical investigation is needed to reduce these challenges. In this study, a modified Deep Convolutional Neural Network (DCNN) architecture using three convolutional layers blocks based on convolution, batch normalization, pooling, fully connected and dropout regularization parameters are employed to hinder overfitting and increase generalization performance is proposed to recognize handwritten digits. Initially, digits taken from the HODA database are pre-processed using various steps including smoothing, black and white images to grayscale intensity images conversion and resizing it to a fixed dimension. Then optimal features extraction and recognition of handwritten images are done by the DCNN algorithm. In deep learning domain, optimization algorithms are considered core solution and their performances highly depends on optimization algorithm selection. In this paper, various optimization algorithms such as stochastic gradient descent (SGD), Adam, Adadelta, Adagrad, Adamax, Momentum, RMSprop and Nag are employed for the optimization of proposed DCNN. Moreover, current research also analyzes the role of different epochs to ameliorate optical character recognition (OCR) performance of Persian/Arabic handwritten digits. At the end, we also worked on finding a suitable composition of learning parameters to establish high performance DCNN architecture that conquer the loopholes of traditional methods. Results reveal that the proposed DCNN model achieves state-of-the-art performance and outperform other studies in the literature.

Keywords OCR · Digit recognition · Deep learning method · Optimization algorithms · HODA database

✉ Saqib Ali
alisaqib@emails.bjut.edu.cn

Extended author information available on the last page of the article.

1 Introduction

Machine learning and deep learning techniques [18, 48] have recently achieved excessive acknowledgment due to a new multidisciplinary domain that attempts to make computer systems more intelligent. The main objective of artificial intelligence is that computer systems can feel, memorize, learn, and identify everything just as human beings do [11, 28]. In many OCR domains such as handwritten Tamil character recognition [24], handwritten Urdu text recognition [22], handwritten digits(0-9) recognition [5, 6] and Chinese handwritten text recognition [49]. In the modern world, images are received and stored digitally. Therefore, (OCR) is obtaining prestige and used in various applications including zip code scanning in post offices, processing cheques in banks, vehicle number plate recognition, etc. At present, automatic recognizing handwritten digits is noteworthy and challenging due to several reasons such as every individual's unique writing style, usage of various writing materials, variations in orientation and size of fonts. Another issue is that a single numeral has multiple writing styles. There should be a model for detecting these handwritten numerals without defining different types for the other writing style. The effort to transform handcrafted or printed characters and digits modifiable by a machine is still an open research challenge for the researchers [19, 23, 37].

OCR technology for handwritten documents of English, Chinese, and Japanese has reached acceptable levels [43]. This technology has not matured due to the difficulties of recognizing handwritten digits in Persian/Arabic and the varied shapes of each digit. The computer must interpret these handwritten digits because they are increasingly used in many sectors of Persian and Arabic states. Persian/Arabic digits scripts are written like English numbers (from left to right). A vital point to be considered for handcrafted digit identification, there are several ways in which digits can be written. Moreover, depending on literacy level and geographical area, individuals write numbers differently. Arabic is the national language of many countries and various nations speaks Arabic language including, Libya,

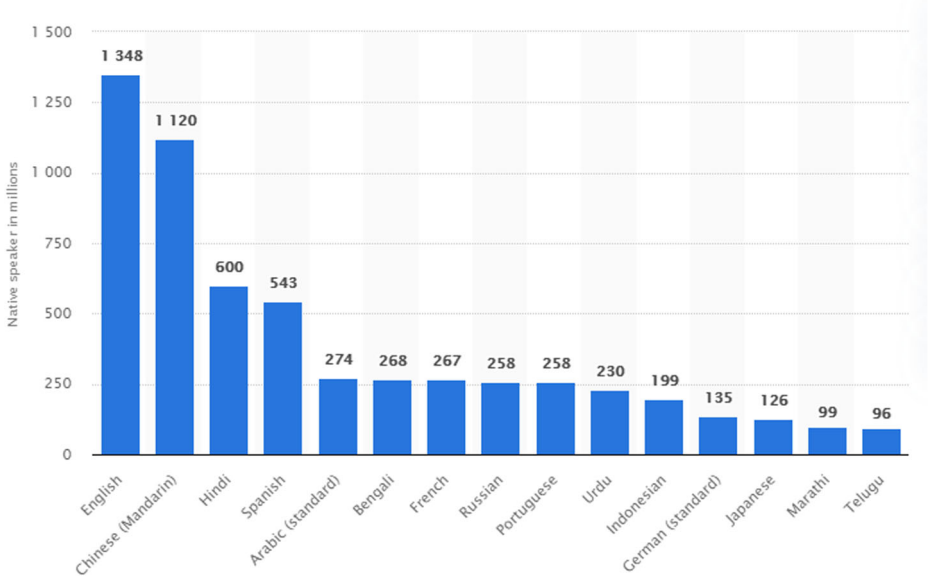

Fig. 1 The statistic showed the most spoken languages worldwide in 2021 [45]

Mauritania, Morocco, Oman, Palestine, Qatar, Saudi Arabia, etc. Altogether nearly 274 million people speak the Arabic language worldwide [45], as illustrated in Fig. 1. Therefore, statistics show the importance of these handwritten digit's scripts.

So far, some investigation has been done in the realm of Persian/Arabic handwritten numeral images. Alaei et al. [3] proposed a recognition system for handwritten numerals using modified contour features. They extract 196 various features for the handcrafted data. Authors used support vector machine (SVM) classifier incorporating Gaussian kernel function to recognize the images. They tested proposed model on 20,000 HODA database samples and obtained 98.71% recognition rate. Zamani et al. [50] proposed a CNN and Random Forest (RF) based Persian handwritten digits recognition model. Cubic interpolation technique was applied for rescaling the handwritten images and few handcrafted features were mined to train the proposed network. The proposed model was compared with other classifiers and implemented deep learning method outperformed in terms of 99.03% recognition rate.

Traditional approaches for handwritten Persian/Arabic digits largely depend on the explicit building of features, for example, geometric and correlation-based ones. Sadri et al. [38] proposed a gradient histogram preprocessing technique, multi-layer perceptron (MLP) neural network was implemented for feature extraction and nearest-neighbors (KNN) classifier used for the classification task. Results show leading recognition accuracy of 98.57% on digits. Parseh et al. [32] used zoning features and hole size collectively from the HODA digits dataset. Principal Component Analysis (PCA) was employed to decrease the dimensions of the features map. The author applied an SVM classifier for digit classification and attained 99.07% recognition accuracy. Karimi et al. [23] implemented ensemble classifiers to classify Persian handwritten digits and extracted 115 apparent features. The best accuracy of 95.28% was achieved from the Tarbiat Modares University (TMU) digits database. Zhan et al. [51] worked on a novel CNN-based method to identify handwritten digit strings applied on two different databases (ORAND-CAR-A and ORAND-CAR-B). Authors use a combination of feature extraction, dimension reduction, and output layers for the input string. Investigated CNN network simulation results demonstrate significant progress in terms of accuracy. The proposed model obtains results on ORAND-CAR-A and ORAND-CAR-B datasets with accuracy rates 92.2% and 94.02%, respectively.

Another [30] investigate Bag of Visual Words (BoVW) combined two classifiers. The hybrid method combines Scale Invariant Feature Transform (SIFT) and Quantum NNs (QNNs) and obtains high accuracy on HODA digits. Safarzadeh et al. [40] proposed a combined model for recognizing the HODA database using the benefits of both CNN and recurrent neural networks (RNN). State-of-the-art performance with 99.37% accuracy was achieved in comparison with others. Safdari and Moein [41] worked on a sparse autoencoder with two layers to extract hierarchical features and used a softmax classifier for classification. The proposed method achieved 98.22% accuracy on HODA digit images. Al-wajih et al. [7] used sliding windows to improve the recognition rates by applying RF and SVM classifiers to HODA digit images and obtained 99.13% accuracy. Kiani et al. [25] suggested restricted Boltzmann machine (RBM) for handwritten digits feature learning. Authors employed spiking neural networks (SNN) and deep belief network (DBN) for classification purposes. Suggested model obtained recognition rate of 95% on HODA digit images.

From the literature survey, we analyze that researchers use conventional handcrafted feature extraction methods (SIFT, PCA, MLP etc.) and classification (RF, KNN, SVM, RNN, BOVW, RBM and DBN). Researchers are also struggling in achieving high performances on HODA dataset. The literature study of handwritten digits recognition inspires us to use

more advanced and robust method like DCNN to accomplish high performance by utilizing deep learning benefits. Our contribution are as followes:

- In this paper, our main objective is to automatically and correctly recognize Persian/Arabic handwritten digit images and achieve more improved results by implementing a modified and robust DCNN method.
- This research presents a system for identifying handwritten images. Identifying and recognizing handwritten Persian/Arabic digits deals with many real-world applications.
- Moreover, this study explores the role of different optimization algorithms such as SGD, Adam, Adadelta, Adagrad, Adamax, Momentum, RMSprop and Nag, which are very crucial in enhancing the proposed architecture performance.
- Parameter optimization is another contribution to reach the ideal configuration for the proposed model. For this goal, many hyperparameters have been manually adjusted up to the optimum recognition model.
- DCNN architecture is used through a comprehensive experimental investigation of the learning parameters on the HODA database and achieved 99.50% testing accuracy.

The rest of the manuscript is organized as follows: Section 2 describes the suggested methodology for Persian/Arabic handwriting digits; Section 3 describe the optimization algorithms. In Section 4, experimental results are explained. Section 5 presents the conclusion and future work.

2 Research methodology

2.1 Data acquisition

Data acquisition of handwritten images is the initial step of recognition system. There are many databases available for the recognition of handwritten Persian/Arabic digits. To evaluate the proposed algorithm, we use the first dataset developed for handwritten Persian numbers known as the HODA database [26]. It comprises approximately 80,000 examples with a resolution of 200 Dots per Inch (DPI). 60,000 images are for training (each class has 6000 images), and the remaining 20,000 images are testing samples (each class has around 2000 images). These binary images were extracted from 12,000 registration forms created by B.Sc. and senior high school students. Subclasses of the HODA database are shown in Table 1.

Table 1 Subclasses details of the HODA database

Classes	Total Samples
Total number of examples	102352
Training set	60000
Test set	20000
Remaining examples (Letters)	22352
No. of Predicted Classes	10

Fig. 2 Some examples of HODA dataset

2.2 Pre-processing of data

The nature of the input digits dataset effect the performance of the recognition process [1]. The job of pre-processing step is to improve the standard of input data by removing redundant details from images for the next phases [29, 35]. In this study, publicly published HODA dataset images are fed into our proposed DCNN system for recognition and classification. Dimensions of HODA dataset images are variant; all images must be consistent in size and shape before inserting into the network. For this purpose, each image of the dataset was first converted into a binary matrix applying a median threshold. Furthermore, morphological operation, such as dilation for expanding the bright parts and median filter, was employed for edge smoothing and noise reduction. The pre-processing steps are given below:

- Images are converted to binary matrices.
- Each image's foreground pixels are set to 1, and the background pixels are set to 0.
- Change the pictures size to 40*40, and the current patterns have been set in the center of the pictures. Figure 2 shows some sample images of employed handwritten digits.

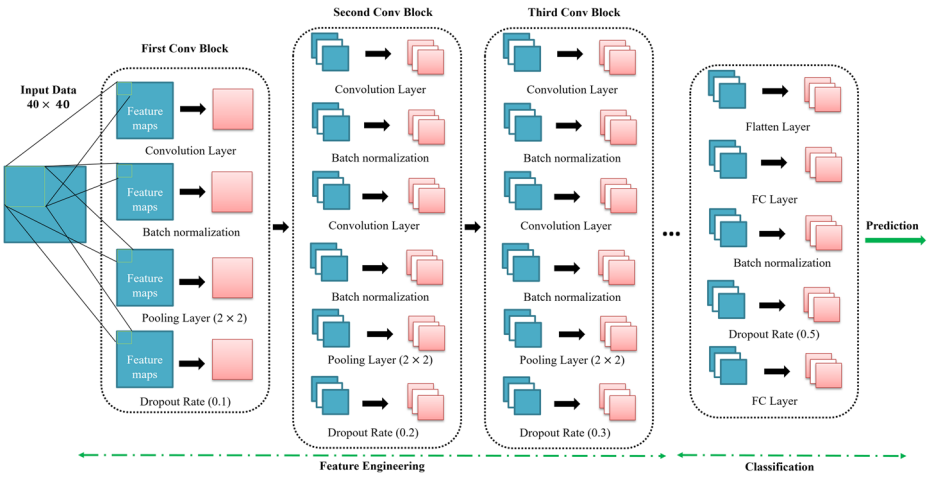

Fig. 3 Proposed DCNN architecture

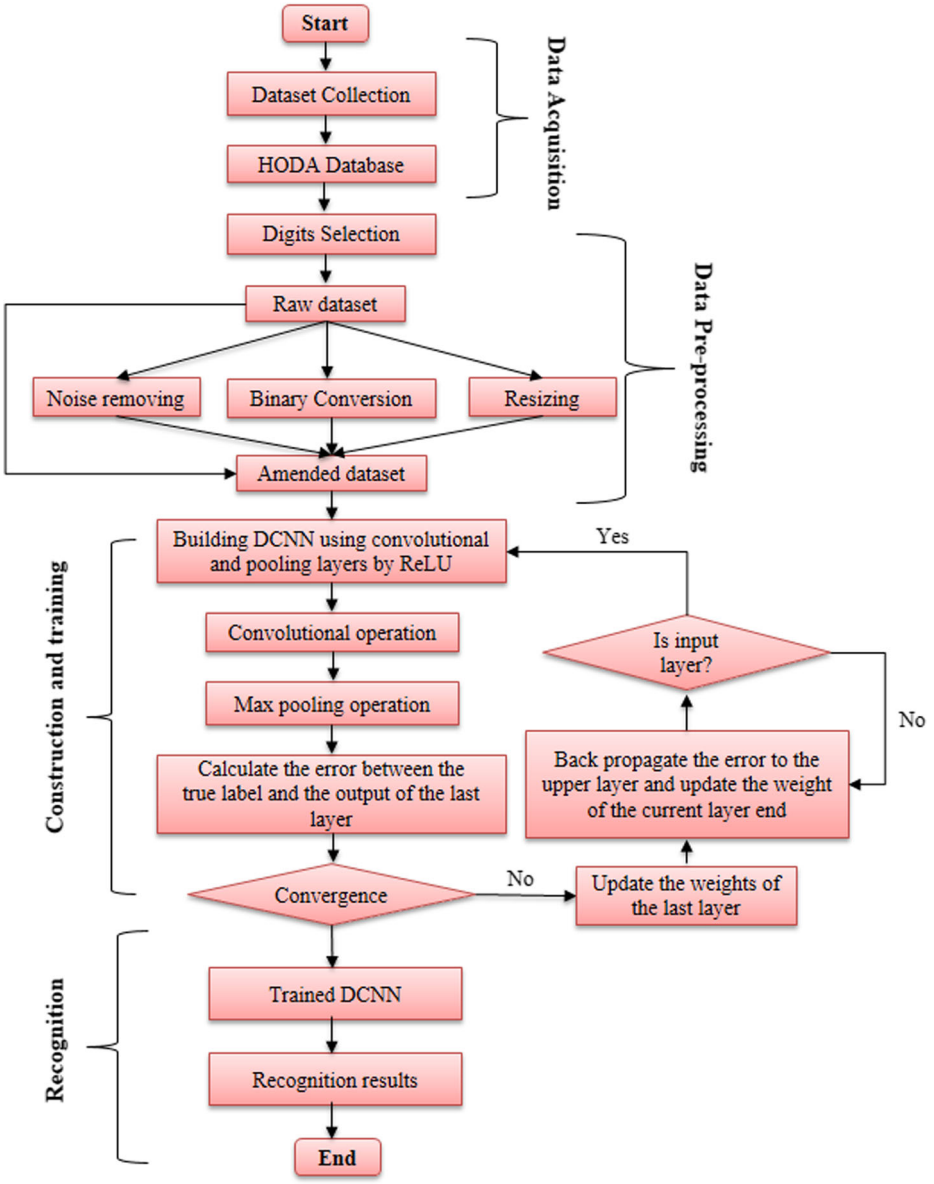

Fig. 4 Flow chart of the proposed method

2.3 Proposed recognition system

This section demonstrate the proposed DCNN architecture design that is constructed via switching convolution layers blocks, batch normalization layers, max-pooling layers, and dropout, followed by FC layers. The primary step for implementing the proposed DCNN system is to apply the recognition method that can recognize Persian/Arabic handwritten numbers with high accuracy. In our research, the suggested DCNN sequential architecture

was employed fundamentally with 23 building blocks, followed by one input layer and one output layer. Figure 3 illustrates a new three Conv. layers building blocks based DCNN architecture of handwritten recognition system. DCNN's are often employed to categorize datasets of larger dimensionalities. Handling large image dimensions is usually tedious. Therefore, DCNN reduces the sample size of any image while preserving the image's spatial resolution. The general framework of the proposed DCNN method is shown in Fig. 4, which includes four main phases namely data acquisition, data pre-processing, construction and training and recognition.

2.3.1 Methodology design for proposed DCNN

Choosing a appropriate design of CNN based deep learning architecture can be done by "hit and trail" rule because every image dataset needs specific adjustments to perform well. The learning process performance is critically delicate to the chosen network model design [4]. Hence, the performance of CNN based approaches depends on the design of structure, input data design and the training procedure controlled by various hyper-parameters. Hyper-parameters including number of filters, layers type, total units per layer, size of stride, pooling size and locations, dropout values, batch normalization, learning rates and depth of the architecture should be carefully tuned for getting improved results. Determining the suitable hyper-parameters combinations for a particular job is very hard due to the lack of clarity of their reaction and impact of interaction on network results. Currently, researcher are using two CNN network design namely the hand-crafted design method and the automated design method [2, 44].

In this study, we have designed our proposed architecture from scratch by applying the hand-crafted design method. We begin with constructing a normal CNN with 1 block, and then we got deeper, using different hyper-parameters combinations for the better performance. In order to achieve the goal of better-generalized and well-performed high and competitive performance, we do more attempts and time to well-tune the hyper-parameters process which leads us to build 4 blocks of Conv. In all convolutional layers of the DCNN network, the actual input representation remain intact by using identical padding. Block 1 of the convolutional features extraction started with the first convolutional layer that possesses 32 feature maps, each with a trainable kernel size of (3*3) pixels and a Rectified Linear Unit(ReLU) activation function of the neurons. It can extract features from the input raw image of size 40*40. Batch normalization layer is second layer in first block which was attached right after the convolutional layers. A 2*2 sized pooling layer was used as the 3rd layer for decreasing the feature maps dimensionality. To reduce the overfitting a dropout layer was employed after pooling layers in each block. Similarly, 3 more convolution layers blocks are stacked to complete the model configuration.

After the completion of convolutionals blocks, we initiate the classification layers block of the network started with flattened layer to convert the 2-dimensional features matrix into a single vector; eventually, it was fed into the 1st FC layer with 1024 neurons. Afterwards one batch normalization layer was connected with a dropout value 0.5. Lastly, a 2nd FC layer was applied at the end of the network. The hidden layers are considered the backbone of any network. The setting of all layers parameters of implemented DCNN architecture for handwritten numerals is represented in Table 2. The individual explanation of the hidden layers(input, convolutional, pooling, FC and output) of our proposed model is as follows:

Table 2 Layers parameters details of the employed DCNN architecture

Block	No of Filter	Stride	Layers	Filter Size
Image Input	-	-	40*40*1	-
Feature Extraction				
1st Conv-Block	64	1*1	Conv Layer	3*3
	-	1*1	Activation Layer (Relu)	-
	-	-	Batch Normalization	-
	-	2*2	Pooling Layer	3*3
	-	-	Dropout (0.1)	-
2nd Conv-Block	128	1*1	Conv Layer	3*3
	-	1*1	Activation Layer (Relu)	-
	-	-	Batch Normalization	-
	128	1*1	Conv Layer	3*3
	-	1*1	Activation Layer (Relu)	-
	-	-	Batch Normalization	-
	-	2*2	Pooling Layer	3*3
	-	-	Dropout (0.2)	-
3rd Conv-Block	256	1*1	Conv Layer	3*3
	-	1*1	Activation Layer (Relu)	-
	-	-	Batch Normalization	-
	256	1*1	Conv Layer	3*3
	-	1*1	Activation Layer (Relu)	-
	-	-	Batch Normalization	-
	-	2*2	Pooling Layer	3*3
	-	-	Dropout (0.3)	-
4th Conv-Block	512	1*1	Conv Layer	3*3
	-	1*1	Activation Layer (Relu)	-
	-	-	Batch Normalization	-
	512	1*1	Conv Layer	3*3
	-	-	Batch Normalization	-
	-	2*2	Pooling Layer	3*3
	-	-	Dropout (0.4)	-
Classification Block				
	-	-	Flatten Layer	-
	-	-	FC Layer (1024)	-
	-	-	Batch Normalization	-
	-	-	Dropout (0.5)	-
	-	-	FC Layer (10 classes)	-
	-	-	Output	-

2.3.2 Input layer

The input images are initially stored in the first layer, known as the input layer. It defines the dimensions (height, width) of the input data and the number of channels (RGB information).

2.3.3 Convolutional layer

The convolutional layer utilizes multiple filters to extract the essential features of an image. The dot product of the input neuron m*m and the filter n*n convoluted through the input image is calculated using (m-n+1).(m-n+1) formula. ReLU activation function used in the convolution layer. Images are often inconsistent; to remove these inconsistencies, the ReLU function is used to form a coherent set with the execution function (convolutional operation).

2.3.4 Max-pooling layer

A pooling layer is also known as a sub-sampling layer. It is placed between two convolutional layers to decrease the dimensionality of the image. Additionally, pooling operations minimize the computational complexity of the network. Suppose an image of W*H*D dimensionality is employed as input in the convolutional layer; the output is a W1*H1*D-size activation map.

$$W1 = \frac{W - F_w}{s} + 1, \tag{1}$$

$$H1 = \frac{H - F_H}{s} + 1. \tag{2}$$

Here W and H are belongs to the image width and height. Filter width and height is denoted by F_W and F_H respectively. W1 and H1 represent the width and height of the activation map, and S is the average value or stride value. The pooling layer also supports the feature selection process and in reducing overfitting. In pooling, the extracted features are used in a specific region and can also be used in other areas. Usually, a 2*2 filter size is utilized for max-pooling operation. The combination of convolutional and pooling layers is used repeatedly until one finally works with a vector that can handle it. Through this pooling approach, activation with the extreme value is adopted from all the activations that exists in a rectangular field, as shown in Fig. 5.

2.3.5 Fully connected layer

It is also called the classification layer, applied as the last layer in DCNN architecture. The input vector obtained after successive conversion and pooling is provided in the final section

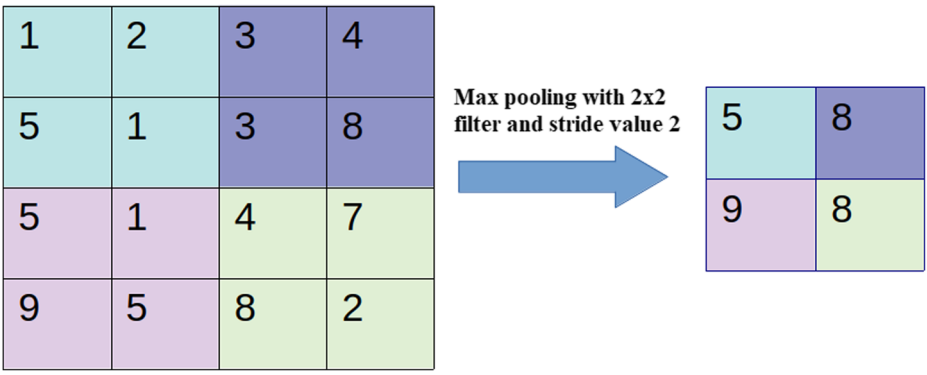

Fig. 5 Representation of max-pooling operation

of the fully connected for classification of handwritten numbers. Softmax function computes predicted classes by recognizing the input image, which is completed by integrating all the features extracted by prior layers. The classification layer applies the Softmax activation function to identify the ten classes generated features based on the training data in our work.

2.3.6 Output layer

The output layer, also known as the dense layer particularly used for prediction. According to the HODA dataset, the total number of output classes are 10, which means we have ten output neurons, representing one digit (0-9).

3 Optimization algorithms

Optimization algorithms are employed to generate effective and efficient results by optimizing the neural network to find the learning parameters' optimal value. The algorithm works by rationalizing the weight/bias values by reducing or maximizing the cost function, known as the network's learning parameters. These learning parameters play a significant role in the generation of a systematic network model. The algorithm upgrading the values is known as the adaptive learning algorithm. Classification experiments in the deep learning field showed outstanding performance, credited to advancement in learning parameters [8, 12, 27, 39, 42]. To resolve the dilemma of loss minimization, CNN employed gradient descent to minimize the error rate throughout the training task and for inner parameters amelioration.

3.1 Stochastic gradient-based optimization

Gradient descent, as the name signifies it manipulates to descend the error gradient alongside the error surface. It is categorized into three different classes: SGD, batch gradient descent, and mini-batch gradient descent. The batch gradient descent is a computationally slow process as it begins to determine the whole training data gradient. Therefore, the best of all choices is SGD. This optimization method is first-order and stochastic, i.e., the update is based on objective value, gradient only, and the subset of training examples. In SGD, parameter update for each training examples a^i and b^i labels were performed. The SGD has numerous advantages over other gradient descent. It limits the cost work and avoids unnecessary computation by allowing the users to customize the vast dataset's algorithm measures. In our experimentation, linear regression is performed employing gradient descent. The mathematical expression is as follows:

$$\theta = \theta - \eta.\nabla_\theta J(\theta; a^i; b^i). \tag{3}$$

Herein, a^i, b^i as training data η as the learning rate, and θ represents the network parameters. Eventually, by following the gradient direction in the space value of the θ parameter, we are following the path that reduces error. Whereby X_E is an error function. Empirically, by reducing the error function (X_E), the best value of θ is manipulated in SGD. Moreover, for linear regression, θ needs the model weight parameters. Thus, the error of the model is $X_E(\theta)$.

3.2 Nesterov Accelerated Gradient (NAG)

NAG optimization gives prescience to the momentum term. It is similar to the spherical ball rolling down a hill; the ball has the notion of its running position to slow down prior to elevation in the gradient of a hill (1). This implies that NAG deals with the gradient calculation with respect to the estimated future position of parameters but not with the current parameter (θ) described in (4) and (5).

$$\upsilon_t = \gamma \upsilon_t - 1 + \eta \nabla_\theta J(\theta - \gamma \upsilon_t - 1), \tag{4}$$

$$\theta = \theta \upsilon_t. \tag{5}$$

Herein, we will use the momentum term to move the θ parameter. Through this, we can effectively estimate the future position of the parameters simply by computing $\theta - \gamma \upsilon_t - 1$. NAG works antagonistically to momentum in terms of calculation of gradient. Owing to looking ahead in the future, faster optimization of decent can be done with NAG. This is why the NAG gradient works slightly better than standard momentum and increases responsiveness in the results. In our experimentation, we have set the momentum term value to 0.9.

3.3 Momentum

Primarily, SGD is one of the accessible optimization methods. However, the time taken to train the model is comparatively higher. Thus, to carry out faster convergence, SGD is optimized with other optimizers such as adadelta, Adam, and momentum. When we talk about momentum, the ball moving down a hill continues to assemble momentum, accelerating on its way until the terminal velocity reaches. The same phenomenon happens in the case of our updated parameter. The dimensions of momentum expression increase in the relevant direction (in the direction of gradient point) and decrease for dimensions whose gradient change directions. It is done by adding the past time fraction of the update vector to the current update vector. In this way, redundant parameter update reduction takes place. As a consequence, faster convergence takes place with the momentum factor (γ).

$$\upsilon_t = \gamma \upsilon_t - 1 + \eta \nabla_\theta J(\theta), \tag{6}$$

$$\theta = \theta - \upsilon_t, \tag{7}$$

The update rule for the momentum is:

$$\nu_t = \mu \nu_{t-1} - \alpha \nabla L_t(\omega_{t-1}), \tag{8}$$

$$\omega_t = \omega_{t-1} + \nu_t. \tag{9}$$

$\alpha > 0$ - learning rate (common value choices: 0.1, 0.01) $\mu \in [0,\ 1)$ - momentum (common value choices: 0.9, 0.95, 0.99) Momentum smooth updates, enhancing stability and speed.

3.4 Adaptive Gradient (Adagrad)

Adagrad algorithm is primarily utilized to optimize sparse data as it performs larger updates for inconsistent data and antagonistically for consistent data [13, 33] by adjusting the learning rate according to the parameters [27]. Hitherto, all parameters (θ) are updated at once as each parameter utilizes the indistinguishable learning rate (η). However, this algorithm uses a different modified learning rate for each parameter (θ_i) with each time step (t) based on past gradient. That why this optimizer is well-suited for sparse data optimization. Herein, we will describe the per parameter update of adagrad followed by vectorization (9). For

simplicity, gt,i refers to the gradient of the objective function at each time step (t) for the parameter (θ_i).

$$g_{t,1} = \nabla_{\theta_t} J(\theta_{t,1}), \tag{10}$$

The updated term at each step time (t) for each parameter (θ_i) can be written as:

$$\theta_{t+1,\,i} = \theta_{t,\,i} - \eta . g_{t,\,i}, \tag{11}$$

Thus, adagrad update rule is:

$$\theta_{t+1,\,i} = \theta_{t,\,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} . g_{t,i}, \tag{12}$$

In this equation $g_{t,i}$ is a gradient of the loss function for parameter θ_i at a time step t, $G_t \in R^{d^* d}$ is a diagonal matrix where each diagonal element i. i is the sum of the squares of the gradients w.r.t. θ_i up to time step t, ϵ is a smoothing term that avoids division by zero (usually on the order of 10^8. According to its update rule, adagrad perform modification at each time step t on learning rate (η) for each and every parameter θ_i based on the past gradients. Moreover, Gt consists of summating the square of a past gradient with respect to all θ parameters diagonally. Hence, we can now perform vectorization by an element-wise matrix-vector multiplication θ between G_t and g_t:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} . g_t. \tag{13}$$

The major advantage of adagrad algorithm is that it does not require manual tuning of the learning rate (η). The default value of 0.01 is utilized in most implementations. Nevertheless, the main drawback is shrinkage or lowering of learning rate due to augmentation of squared gradients in the denominator. This makes the algorithm to compute at its worst.

3.5 Adaptive delta (Adadelta)

To overcome the decaying learning rate problem encountered in adaGrad new learning algorithm was introduced known as Adadelta [53]. It limits the accumulation of past squared gradient as in adagrad by setting the window size. The learning rate is generally the square root of the ratio between moving averages of gradient square and weight update square. Here, the running average is denoted as $E[g^2]_t$ and it depends on current gradient and previous average.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2; \tag{14}$$

The value of γ is adjusted around 0.9, the same as the value of momentum. Now the reformed equation SGD update with respect to parameter update vector $\triangle\theta_i$ can be written as (14):

$$\triangle\theta_i = -\eta . g_{t,i}, \tag{15}$$

$$\theta_{t+1} = \theta_t + \triangle\theta_t, \tag{16}$$

Similarly, parameter update vector $\triangle\theta$ of adaGrad (12) can be written as:

$$\triangle\theta_t = -\frac{\eta}{\sqrt{G_t + \varepsilon}} . g_t, \tag{17}$$

Moreover, the diagonal matrix term G_t in the above equation is now replaced with $E[g^2]t$. Thus the mathematical form would be:

$$\triangle\theta_t = -\frac{\eta}{\sqrt{E[g^2]t + \varepsilon}} . g_t, \tag{18}$$

$\sqrt{E[g^2]t+\varepsilon}$ refers to the root mean squared (RMS) error of gradient. Thus, for simplicity, we can write the expression as:

$$\triangle\theta_t = -\frac{\eta}{RMS[g]_t}.g_t. \tag{19}$$

3.6 Adaptive moment estimation (Adam)

Adam is a versatile algorithm in efficient computation, minute memory, and first-order gradient. Moreover, gradient rescaling is insensitive to the variation in the magnitude of parameter updates. It deals with large-scale high-dimensional machine learning problems. This algorithm amalgamates the advantage of both methods: RMSprop [47] and adaGrad [14], which efficiently perform optimization with non-stationary objects and sparse gradients, respectively. This technique individually computes the adaptive learning rates of various parameters from 1st and 2nd order moments.

Adam works by keeping exponential moving averages of gradient (s_t) similar to RMSProp and its square (d_t), just like Adadelta. Moreover, its update is proportional to $\frac{averagegradient}{averagesquaredgradient}$

$$r_t = \nabla_\theta f_t(\theta_{t-1}), \tag{20}$$

$$s_t = w_1.s_{t-1}(1-w_1).r_t, \text{ (update biased 1st momentum estimate)} \tag{21}$$

$$d_t = w_2.d_{t-1}(1-w_2).r_t^2, \text{ (update biased 2nd momentum estimate)} \tag{22}$$

$$\widehat{s_t} = \frac{s_t}{(1-w_1^t)}, \text{ (bias 1st momentum)} \tag{23}$$

$$\widehat{d_t} = \frac{d_t}{(1-w_2^t)}, \text{ (bias 2nd momentum)} \tag{24}$$

$$\theta_t = \theta_{t-1} - \alpha\frac{\widehat{s_t}}{(\sqrt{\widehat{d_t}+\epsilon})}, \text{ (update parameter)} \tag{25}$$

Here, $f(\theta)$ is the stochastic scalar function that is differentiable w.r.t parameter θ, $r_t = \nabla_\theta f_t(\theta)$denote gradient, i.e., the vector of partial derivatives of f_t w.r.t θ evaluated at time step t, s_t is the algorithm update exponential moving averages of gradient, s_t= algorithm update exponential squared moving averages of gradient, w_1 and w_2 are hyper parameters where, $w_1, w_2 \in [0,\ 1)$ controls the exponential decay rates of these moving averages. Using update the parameters just as we have seen in Adadelta and RMSprop, which yields the Adam update rule:

$$\alpha_t = \alpha.\frac{\sqrt{1-\omega_2^t}}{1-\omega_1^t}, \tag{26}$$

$$\theta_t = \theta_{t-1} - \alpha_t.\frac{s_t}{\sqrt{\widehat{d_t}+\widehat{\varepsilon}}}. \tag{27}$$

3.7 Adaptive max pooling (Adamax)

In Adam update rule, the factor d_t computes the gradient, which is inversely proportional to the l2 norm of $|g_t|^2$ (current gradient) and past gradient (υ_{t-1}).

$$\upsilon_t = \beta_2\upsilon_{t-1}(1-\beta_2)|g_t|^2, \tag{28}$$

We generalize this equation with respect to lp norm

$$\upsilon_t = \beta_2^\rho\upsilon_{t-1}(1-\beta_2^\rho)|g_t|^\rho, \tag{29}$$

The norm for small ρ values works stably, but it becomes numerically unstable when the p-value becomes large. This is why l1 and l2 norms are in common practice. Nevertheless, 1_∞ also show stable behavior. For this reason, adamax was discovered by the authors as it utilizes 1_∞ with υ_t to perform convergence. The result exhibits more stable value. The infinity norm constrained υ_t is denoted by the simple term μ_t. This μ_t also remove confusion with Adam.

$$\upsilon_t = \beta_2^\infty \upsilon_{t-1}(1-\beta_2^\infty)|g_t|^\infty = \max(\beta_2.\upsilon_{t-1}, |g_t|), \tag{30}$$

The adamax update rule is:

$$\theta_{t+1} = \theta_t \frac{\eta}{\upsilon_t}\widehat{m}_t, \tag{31}$$

This equation shows that by replacing with $\sqrt{\upsilon_t + \epsilon}$ Adam update rule is changed to adamax update rule.

$$\theta_{t+1} = \theta_t \frac{\eta}{\sqrt{\upsilon_t + \epsilon}}\widehat{m}_t. \tag{32}$$

In the adamax algorithm, there is no need for bias correction of μ_t as in Adam optimization. The excellent default value of $\eta = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ is utilized in most implementations.

3.8 Root mean square propagation (RMSProp)

RMSprop is another adaptive learning method anticipated by Geoff Hinton. This optimization technique is closely related to Adam, RMSprop, and adgrad. RMSProp works in combination with momentum by utilizing momentum and generates parameter updates on rescaled gradient. RMSprop lacks bias-correction term as in Adam. Moreover, to overcome the drawback that encounters in the adaGrad algorithm, i.e., radically decreasing learning rate with time (t) RMSprop has been developed. RMSprop is more or less resembles the first update vector of adaGrad. However, it is the scale by decaying average (avg.) of squared gradient rather than the sum of the squared gradient in adaGrad.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2, \tag{33}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}}g_t. \tag{34}$$

The equation explains well that RMSprop computes by dividing the learning rate with an average of squared gradients decaying exponentially. The default value of the learning rate in which this algorithm works well in our work is 0.001.

4 Results and discussion

For the implementation of the proposed work, we used a tensorflow integrating keras library using python language. In training, we utilized pycharm tool and tensorflow integrating keras library to implement the proposed DCNN architecture. NVIDIA Geforce GTX1660Ti 4gb GPU with 8 GB RAM. The model is trained using 100 epochs size and size of an input HODA data is 40*40*40. The proposed DCNN-based scheme through additional convolutional blocks (i.e convolutional layers, Relu activation function, batch normalization, pooling layers and dropout regularization) incorporated fully connected layers was applied for 100 epochs to automatically recognize and correctly classify Persian/Arabic handwritten number images.

4.1 Evaluation measures

This segment of the paper briefly discusses the performance evaluation measures of the proposed DCNN model. All exploited multi-class performance evaluation metrics are defined below:

4.1.1 Accuracy

Accuracy [34] is the percentage of truly predicted examples from the entire predictions made by the algorithm. Mathematically, it can be defined as:

$$Accuracy = \frac{Tp + Tn}{Tp + Fp + Tn + Fn} \tag{35}$$

where, Tp (True positive): model classifies a positive sample as positive, Tn (True negative): model classifies a negative sample as negative, Fp (False positive): model misclassifies a negative sample as positive, Fn (False negative): model misclassifies a positive sample as negative.

4.1.2 Precision

Precision [20] metrics estimate how many examples predicted as positive by the implemented algorithm truly belong to the positive class. It can be calculated as:

$$Precision = \frac{Tp}{Tp + Fp} \tag{36}$$

4.1.3 Recall

Recall [20] measures estimate which section of samples that really belong to the positive class is truly predicted positively by the developed architecture. Mathematically, it can be written as:

$$Recall = \frac{Tp}{Tp + Fn} \tag{37}$$

4.1.4 F1 score

F1-score [20] is calculated by taking the harmonic mean of precision and recall. It can be defined as:

$$F1 - score = \frac{2 * p * r}{p + r} \tag{38}$$

4.2 DCNN-based handwritten digit recognition

We have trained and tested the DCNN model on the HODA dataset. The training and testing recognition accuracies for handwritten digits are repersented in the form of bar grapghs in Fig. 6(a). A maximum training recognition rate, i.e., 99.90%, was observed for numbers 0, 5 and 7. The least training accuracy was found at number 4(99.55%). Similarly, maximum test accuracy was observed for number 0(99.70%), while the least test accuracy was found at number 5(99.20%). Comprehensively, 99.80% training accuracy on the training dataset and 99.50% test recognition accuracy were observed, promising and surpassing

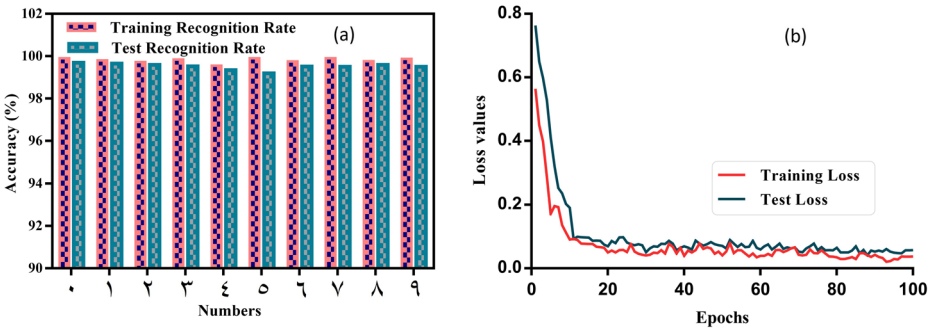

Fig. 6 (a) Training and test recognition rate (b) Training and test loss

the ones obtained in the prior research recognition rates. Furthermore, Fig. 6(b) is representing training and testing dataset losses against the number of epochs. We can see that proposed model effectively achieving the high testing accuracy and the lowest testing loss. Loss is decreasing continuously as epochs are increasing for both training and testing samples. The generalization gap (accuracy and losses) between training and test dataset should be as narrow as possible to avoid overfitting the model. We also assess our proposed DCNN technique performance through different metrics, including accuracy, precision, recall and F1-score. Table 3 represents complete detail of various measures reported against each persian handwritten number scripts for testing the HODA dataset.

4.3 Role of optimization algorithms

Furthermore, to validate the suggested model effectiveness, we have applied different gradient descent optimizers (see Tables 4, 5, 6 and 7. Most favorable parameters were used to accomplish the overall task (i.e., Persian/Arabic handwritten digit recognition). We implemented up to 100 epochs to achieve state-of-the-art results on handwritten digit images. In the current study, eight different gradient descent optimizers and a different number of

Table 3 Recognition results on HODA test dataset handwritten images

Digits	Precision	Recall	F1-score	Accuracy
0	0.997	1	0.999	0.997
1	0.998	0.999	0.996	0.996
2	0.995	0.996	0.996	0.996
3	0.994	0.998	0.997	0.995
4	0.993	0.985	0.988	0.993
5	0.993	0.982	0.987	0.992
6	0.994	0.991	0.993	0.995
7	0.995	1	0.996	0.995
8	0.996	1	0.997	0.996
9	0.995	0.999	0.996	0.995
Total	-	-	-	0.995

Table 4 Proposed DCNN model training accuracy of various optimizers

Optimizers→ Epoch ↓	Adam	Ada-delta	Ada-grad	Ada-max	Mom-entum	RMS prop	SGD	Nag
20	98.0	96.0	96.0	95.0	97.0	95.5	95.0	94.0
40	98.5	97.5	97.0	95.0	97.2	95.5	95.4	94.0
60	98.5	97.5	97.0	96.0	97.2	95.6	96.1	94.4
80	99.2	97.8	97.0	96.0	97.4	95.9	96.7	94.4
100	99.8	97.8	97.0	96.0	97.4	96.0	97.0	94.7

Table 5 Proposed DCNN model test accuracy of various optimizers

Optimizers→ Epoch ↓	Adam	Ada-delta	Ada-grad	Ada-max	Mom-entum	RMS prop	SGD	Nag
20	98.0	97.0	96.0	95.0	96.8	95.5	95.0	94.0
40	98.5	97.5	96.5	95.0	97.2	95.5	95.4	94.0
60	98.6	97.5	96.5	96.0	97.2	95.6	96.5	94.4
80	99.0	97.7	96.8	96.0	97.4	95.9	96.5	94.4
100	99.5	97.7	97.4	96.0	97.5	96.0	96.8	94.7

Table 6 Training loss of various optimizers using HODA dataset

Optimizers→ Epoch ↓	Adam	Ada-delta	Ada-grad	Ada-max	Mom-entum	RMS prop	SGD	Nag
20	0.05	0.06	0.06	0.07	0.06	0.08	0.07	0.09
40	0.04	0.06	0.06	0.07	0.06	0.08	0.07	0.09
60	0.04	0.05	0.06	0.06	0.06	0.07	0.07	0.08
80	0.03	0.05	0.04	0.06	0.06	0.07	0.06	0.08
100	0.03	0.05	0.04	0.05	0.06	0.07	0.05	0.08

Table 7 Test loss of various optimizers using HODA dataset

Optimizers→ Epoch ↓	Adam	Ada-delta	Ada-grad	Ada-max	Mom-entum	RMS prop	SGD	Nag
20	0.07	0.11	0.12	0.14	0.12	0.14	0.13	0.16
40	0.07	0.11	0.11	0.13	0.12	0.14	0.13	0.16
60	0.06	0.10	0.10	0.13	0.11	0.13	0.12	0.15
80	0.06	0.09	0.09	0.12	0.10	0.12	0.11	0.15
100	0.05	0.09	0.09	0.12	0.10	0.12	0.11	0.14

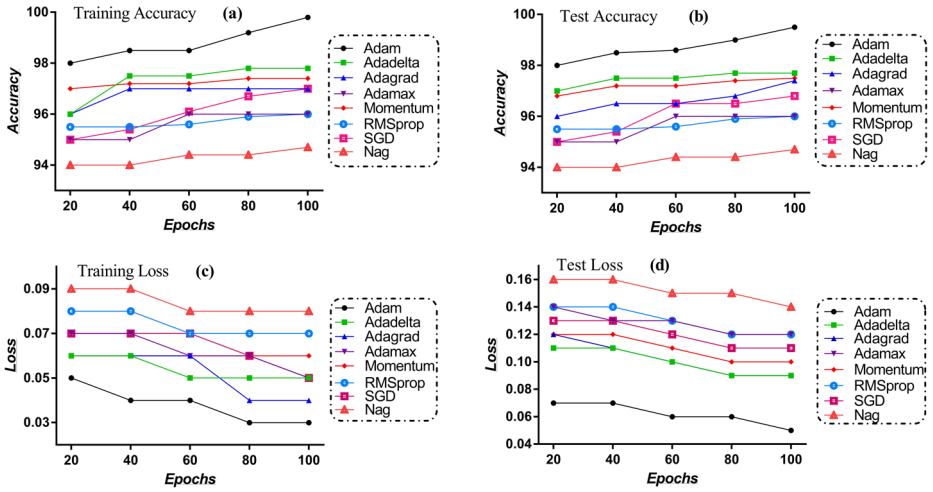

Fig. 7 Comparison of accuracy and loss of DCNN algorithm using different optimizers

epochs (20, 40, 60, 80, and 100) are utilized to evaluate the performance of DCNN. Tables 4, 5, 6 and 7 represent the experimental results of the epochs, as mentioned earlier.

The lowest error rate was observed by using the Adam optimization algorithm in our proposed DCNN model. Adam optimizer's performance was most effective in all experiments carried out to recognize HODA handwritten digits. Although all other optimizers' performances are also sound, adadelta and momentum performances are near to Adam optimizer. Figure 7 illustrates the performances of various optimizers graphically, implying our proposed method with different epochs. In Fig. 7a, we can notice that Adam, adadelta, and momentum impart the top training rates at 100 epochs. Figure 7b, indicates that Adam has the maximum test accuracy while NAG comes up with low test accuracy. Apart from Adam, adadelta, and momentum other optimizers have the ability to achieve high recognition accuracy. Figure 7c, demonstrates that Adam obtained the lowest training loss while RMSProp, SGD, and NAG have leading training losses. In Fig. 7d, we can see that RMSProp, SGD, adamax, and NAG produce the maximum test losses, and Adam has the minimum test loss. After analyzing the experimental results of gradient descent optimizers, we can say that each optimizer produces imbalance recognition results using different epochs for the proposed DCNN design.

4.4 Performance analysis using confusion matrix and existing studies

The HODA handwritten digits classification performance is showed in the confusion matrix as presented in Table 8. HODA dataset contain around 20,000 handwritten images samples were used for testing purpose, only overall 5% test data were misclassified. For the numerals 3, 6, 7 and 9 recognition rate is 0.995 because 0.5 data are wrongly predicted and only 3 images for each digit is misclassified by our model. For the digits 1, 2, and 8 recognition rate is 0.996 because 0.4 data are wrongly predicted and only 2 images for each digit is misclassified by the network. Digit 0 obtained highest recognition rate of 0.997 because only one image is misclassified as 5 and digit 5 obtained lowest recognition rate of 0.992 because 2 images are misclassified as 0 digit, 1 as 4 and 1 as 9 digit. Digit 4 also not perform better and obtained second lowest performance with recognition rate of 0.993. In confusion matrix,

Table 8 Confusion matrix demonstrating the classification results of proposed DCNN model

		Predicted As											
	Digits	0	1	2	3	4	5	6	7	8	9	TPR(%)	FNR(%)
True Input	0	0.997					0.3					0.997	0.3
	1		0.996	0.1	0.1	0.2						0.996	0.4
	2			0.996	0.1	0.3						0.996	0.4
	3			0.4	0.995	0.1						0.995	0.5
	4		0.3			0.993	0.1				0.3	0.993	0.7
	5	0.4				0.2	0.992				0.2	0.992	0.8
	6	0.1	0.1					0.995		0.1	0.2	0.995	0.5
	7		0.2	0.1	0.1			0.1	0.995			0.995	0.5
	8		0.2		0.1					0.996	0.1	0.996	0.4
	9	0.1	0.2					0.2			0.995	0.995	0.5

we have presented that which wrongly predicted image is classified for which class. After attentive investigation of the patterns of HODA handwritten images dataset, the reasons behind these miss classifications are fairly understandable.

The comparison of recognition performance of our proposed DCNN approach with traditional approaches is mentioned in Table 9. Soham De et al. [12] implemented various common auto-encoders on VGG-9 with CIFAR-10 and MNIST to establish that NAG has better capacities to reduce the gradient norms. Mini-batching does seem to help NAG do better than ADAM on small nets. Huizhen Zhao et al. [52] proposed energy index-based optimization method EIOM and compared with various optimization algorithms including AdaGrad, RMSProp, Adam, AdaDelta, and CLR, and achieve 97% accuracy on the CIFAR-10 dataset. However, no one compared optimization algorithms for Persian/Arabic handwritten digits yet notably on HODA datset. In our experiments Adam algorithm perform better then all other optimizers using said database. Adam integrate the splendid features of the AdaGrad and RMSProp optimization algorithms that can control sparse gradients on noisy data. Table 9 shows that the digits from the HODA database specifically, our proposed modified DCNN approach achieved a 99.50% recognition rate for test dataset and outperforms compared to other methods heretofore. The suggested model has low computational complexity and a quick processing speed, taking an average testing time of 0.11 to 0.23 s to recognize and classify a Persian/Arabic handwritten digit, indicating that the model outperforms other traditional DL models. Some existing methods like SVM obtained over 99% accuracy. However these methods used handcrafted feature extraction techniques for instance Mohammad Javad Parseh and Mojtaba Meftahi [32], applied Principal Component Analysis (PCA) method for reducing the feature vector dimensions. Finally, data are classified by Support Vector Machine (SVM) classification method. Ebrahim Al-wajih and Rozaida Ghazali [7], proposed offline Arabic digit recognition using sliding window approach, authors applied four different handcrafted feature extraction techniques includes Mean-based, Gray-Level Co-occurrence Matrix (GLCM), Moment-based, and Edge Direction Histogram (EDH). Authors employed AHDBase dataset to evaluate the performance

Table 9 Recognition accuracy comparison of the proposed method with traditional techniques

Ref	Year	Method	Dataset	Recognition Results (%)
El-Sawy et al. [16]	2016	CNN + LeNet50	ADBase	88
Dehghanian and Ghods [15]	2018	CNN + LeNet	HODA	97.38
Zhao et al. [52]	2019	2D-CNN	CIFAR-10	97.0
Takruri et al.[46]	2014	FCM-SVM	Public database	88.18
Husnain et al.[22]	2019	CNN	UNHD	98.30
Rashnodi et al. [36]	2011	SVM	HODA	98.94
Parseh and Meftahi [32]	2017	SVM	HODA	99.07
Farahbakhsh et al. [17]	2017	AlexNet	HODA	98.75
Ashiquzzaman et al. [10]	2019	CNN	CmaterDb3.3.1	99.4
De et al. [12]	2018	VGG-9	CIFAR-10	99.0
AlKhateeb and Alseid [9]	2014	Bayesian	ADBase	85.26
Al-wajih and Ghazali [7]	2020	SVM	AHDBase	99.13
Hosseini-Pozveh et al. [21]	2020	IT2FRBM	HODA	97.12
Nanehkaran et al. [31]	2021	VGG16	HODA	90.26
Nanehkaran et al. [31]	2021	ResNet18	HODA	93.75
Our	2022	DCNN	HODA	99.50

of the proposed system and use SVM and RF as classification methods. All other studies in Table 9 that uses HODA database not perform well even in term of accuracy. The maximum accuracy was 99.07% on HODA dataset achieved by [32], which is less than the proposed DCNN method. In contrast, our study uses DCNN model which is more efficient and automatically extract important features than classify using softmax function.

5 Conclusion and future work

This paper investigates the deep learning-based system called modified DCNN proposed for handwritten Persian/Arabic digit recognition. The proposed model automatically generates a feature using DCNN and output prediction using the Softmax function. The experimental results demonstrate that, the proposed approach attains the recognition accuracy of 99.50% for the HODA test dataset. Moreover, this study also analyzes several optimization algorithms using different numbers of epochs to ameliorate handwritten digit performance. All optimization algorithms perform well, but experimental results show Adams encouraging performance compared with other optimizers because it combines the finest characteristics of the AdaGrad and RMSProp optimization algorithms that can handle sparse gradients on noisy data. Adams bias-correction helps in achieving best performance over other algorithms. Many testing and experiments have been postponed due to a shortage of hardware resources. As the feature size increases, so it also increases the training time and other requirements. For these reasons, only images of 40*40*40 pixels were utilized in all experiments to reduce the amount of features, while the proposed approach need be confirmed using alternative image sizes. Moreover, the proposed system use only one CNN classifier and validated using one dataset, which may be biased towards the proposed system.

We believe that this study can be helpful to researchers who want to investigate handwritten digits recognition. Furthermore, this research can be beneficial for the researchers

who want to research in image processing, especially on optical character recognition. In the future, Persian/Arabic handwritten digits and characters can also be investigated with appropriate results and apply more state-of-the-art pre-trained models such as MobileNet V2, VGG19, ResNet and GoogleNet.

Acknowledgements Authors would like to thank Ms Rooha Khurram for proofreading the manuscript.

Author Contributions Each author took part in the present work conception and/or design. Tasks of data collection, material preparation, data analysis, and writing of the original draft were executed by Saqib Ali and Sana Sahiba. Muhammad Azeem, Zeeshan Shaukat, Tariq Mahmood, Zareen Sakhawat and Muhaamd Saqlain Aslam helped in reviewing, and editing the manuscript. All authors read and approved the final manuscript.

Declarations

Conflict of Interests The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. Abodi JA, Li X (2014) An effective approach to offline arabic handwriting recognition. Comput Electr Eng 40(6):1883–1901
2. Ahmed AAF, Darwish SMS, El-Sherbiny MM (2019) A novel automatic cnn architecture design approach based on genetic algorithm. In: International conference on advanced intelligent systems and informatics. Springer, pp 473–482
3. Alaei A, Pal U, Nagabhushan P (2009) Using modified contour features and svm based classifier for the recognition of Persian/Arabic handwritten numerals. In: 2009 seventh international conference on advances in pattern recognition, pages 391–394
4. Albelwi S, Mahmood A (2017) A framework for designing the architectures of deep convolutional neural networks. Entropy 19(6):242
5. Ali S, Shaukat Z, Azeem M, Sakhawat Z, Mahmood T et al (2019) An efficient and improved scheme for handwritten digit recognition based on convolutional neural network. SN Applied Sciences 1(9):1–9
6. Ali S, Li J, Pei Y, Aslam MS, Shaukat Z, Azeem M (2020) An effective and improved cnn-elm classifier for handwritten digits recognition and classification. Symmetry 12(10):1742
7. Al-wajih E, Ghazali R (2020) Improving the accuracy for offline arabic digit recognition using sliding window approach. Iranian J Sci Technol, Trans Elect Eng 44(4):1633–1644
8. Alzubi J, Nayyar A, Kumar A (2018) Machine learning from theory to algorithms an overview. In: Journal of physics conference series. IOP Publishing, vol 1142, pp 012012
9. AlKhateeb JH, Alseid M. (2014) Dbn-based learning for arabic handwritten digit recognition using dct features. In: 2014 6th international conference on computer science and information technology (CSIT). IEEE, pp 222–226
10. Ashiquzzaman A, Tushar Abdul K, Rahman A, Mohsin F (2019) An efficient recognition method for handwritten arabic numerals using cnn with data augmentation and dropout. In: Data management, analytics and innovation. Springer, pp 299–309
11. Cireşan D. C., Meier U, Gambardella LM, Schmidhuber J (2010) Deep, big, simple neural nets for handwritten digit recognition. Neural Comput 22(12):3207–3220
12. De S, Mukherjee A, Ullah E (2018) Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration. arXiv:1807.06766
13. Dean J, Corrado SG, Monga R, Chen K, Devin M, Le QV, Le MZL, Ranzato MarcAurelio, Senior A, Tucker P et al (2012) Large scale distributed deep networks
14. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res, vol 12, (7)
15. Dehghanian A, Ghods V (2018) Farsi handwriting digit recognition based on convolutional neural networks. In: 2018 6th international symposium on computational and business intelligence (ISCBI). IEEE, pp 65–68

16. El-Sawy A, Hazem EL-Bakry, Loey M (2016) Cnn for handwritten arabic digits recognition based on lenet-5. In: International conference on advanced intelligent systems and informatics. Springer, pp 566–575
17. Farahbakhsh E, Kozegar E, Soryani M (2017) Improving persian digit recognition by combining data augmentation and alexnet. In: 2017 10th iranian conference on machine vision and image processing (MVIP). IEEE, pp 265–270
18. Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena-Martinez V, Garcia-Rodriguez J (2017) A review on deep learning techniques applied to semantic segmentation. arXiv:1704.06857
19. Hamidi M, Borji A (2010) Invariance analysis of modified c2 features case study handwritten digit recognition. Mach Vis Appl 21(6):969–979
20. Hossin M, Md NS (2015) A review on evaluation metrics for data classification evaluations. Int J Data Mining Knowl Manag Process 5(2):1
21. Hosseini-Pozveh MS, Safayani M, Mirzaei A (2020) Interval type-2 fuzzy restricted boltzmann machine. IEEE Trans Fuzzy Syst 29(5):1133–1142
22. Husnain M, Missen MMS, Mumtaz S, Jhanidr MZ, Coustaty M, Luqman MM, Ogier J-M, Choi GS (2019) Recognition of urdu handwritten characters using convolutional neural network. Appl Sci 9(13):2758
23. Karimi H, Esfahanimehr A, Mosleh M, Salehpour S, Medhati O et al (2015) Persian handwritten digit recognition using ensemble classifiers. Proc Comput Sci 73:416–425
24. Kavitha BR, Srimathi C (2019) Benchmarking on offline handwritten tamil character recognition using convolutional neural networks. J King Saud Univ-Comput Inf Sci
25. Kiani K, Mohsenzadeh KE (2015) Classification of persian handwritten digits using spiking neural networks. In: 2015 2nd international conference on knowledge-based engineering and innovation (KBEI). IEEE, pp 1113–1116
26. Khosravi H, Kabir E (2007) Introducing a very large dataset of handwritten farsi digits and a study on their varieties. Pattern Recognit Lett 28(10):1133–1141
27. Kingma DP, Adam BJ (2014) A method for stochastic optimization. arXiv:1412.6980
28. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
29. Mohamad DNM-A, Hassan H, Haron H (2015) A review on feature extraction and feature selection for handwritten character recognition. Int J Adv Comput Sci Appl 6:204–212
30. Montazer GA, Soltanshahi MA, Giveki D (2017) Farsi/arabic handwritten digit recognition using quantum neural networks and bag of visual words method. Optical Memory and Neural Networks 26(2):117–128
31. Nanehkaran YA, Chen J, Salimi S, Zhang D (2021) A pragmatic convolutional bagging ensemble learning for recognition of farsi handwritten digits. Journal Supercomput 77(11):13474–13493
32. Parseh MJ, Meftahi M (2017) A new combined feature extraction method for persian handwritten digit recognition. Int J Image Graphics 17(02):1750012
33. Pennington J, Socher R, Manning CD (2014) Glove Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
34. Protopapadakis E, Doulamis A, Doulamis N, Maltezos E (2021) Stacked autoencoders driven by semi-supervised learning for building extraction from near infrared remote sensing imagery. Remote Sens 13(3):371
35. Qacimy BE, Hammouch A, Kerroum MA (2015) A review of feature extraction techniques for handwritten arabic text recognition. In: 2015 International conference on electrical and information technologies (ICEIT). IEEE, pages 241–245
36. Rashnodi O, Sajedi H, Abadeh MS (2011) Using box approach in persian handwritten digits recognition. Int J Comput Appl 32(3):1–8
37. Salimi H, Giveki D (2013) Farsi/arabic handwritten digit recognition based on ensemble of svd classifiers and reliable multi-phase pso combination rule. Int J Doc Anal Recognit (IJDAR) 16(4):371–386
38. Sadri J, Yeganehzad MR, Saghi J (2016) A novel comprehensive database for offline persian handwriting recognition. Pattern Recogn 60:378–393
39. Saeed F, Paul A, Karthigaikumar P, Nayyar A (2019) Convolutional neural network based early fire detection. Multimed Tools Appl:
40. Safarzadeh VM, Jafarzadeh P (2020) Offline persian handwriting recognition with cnn and rnn-ctc. In: 2020 25th international computer conference, computer society of Iran (CSICC), pages 1–10
41. Safdari R, Moin M-S (2016) A hierarchical feature learning for isolated farsi handwritten digit recognition using sparse autoencoder. In: 2016 artificial intelligence and robotics (IRANOPEN). IEEE, pages 67–71

42. Shi Z, Ye Y, Yunpeng Wu (2016) Rank-based pooling for deep convolutional neural networks. Neural Netw 83:21–31
43. Srihari SN, Ball G (2012) An assessment of arabic handwriting recognition technology. In: Guide to OCR for Arabic scripts. Springer, pp3–34
44. Sun Y, Xue B, Zhang M, Yen GG (2019) Completely automated cnn architecture design based on blocks. IEEE Trans Neural Netw Learn Syst 31(4):1242–1254
45. Szmigiera M (2021) https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/. Accessed 10 Dec 2020, 03 Jan 2021, 16 Aug 2022
46. Takruri M, Al-Hmouz R, Al-Hmouz A (2014) A three-level classifier: fuzzy c means, support vector machine and unique pixels for arabic handwritten digits. In: 2014 world symposium on computer applications & research (WSCAR). IEEE, pp 1–5
47. Tieleman T, Hinton G et al (2012) Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA Neural Netw Mach Learn 4(2):26–31
48. Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E (2018) Deep learning for computer vision: a brief review. Comput Intell Neurosci:2018
49. Xie Zecheng, Sun Z, Jin L, Feng Z, Zhang S, (2016) Fully convolutional recurrent network for handwritten chinese text recognition. In: 2016 23rd international conference on pattern recognition (ICPR). IEEE, pp 4011–4016
50. Zamani Y, Souri Y, Rashidi H, Kasaei S (2015) Persian handwritten digit recognition by random forest and convolutional neural networks. In: 2015 9th Iranian conference on machine vision and image processing (MVIP). IEEE, pp 37–40
51. Zhan H, Lyu S, Lu Y (2018) Handwritten digit string recognition using convolutional neural network. In: 2018 24th international conference on pattern recognition (ICPR). IEEE, pp 3729–3734
52. Zhao H, Liu F, Zhang H, Liang Z (2019) Research on a learning rate with energy index in deep learning. Neural Netw 110:225–231
53. Zhijian Qu, Yuan S, Chi R, Chang L, Zhao L (2019) Genetic optimization method of pantograph and catenary comprehensive monitor status prediction model based on adadelta deep neural network. IEEE Access 7:23210–23221

Affiliations

Saqib Ali[1] · Sana Sahiba[1] · Muhammad Azeem[2] · Zeeshan Shaukat[1] · Tariq Mahmood[3] · Zareen Sakhawat[1] · Muhammad Saqlain Aslam[4]

Sana Sahiba
sana.sahiba@outlook.com

Muhammad Azeem
mazeem.qau@hotmail.com

Zeeshan Shaukat
zee@emails.bjut.edu.cn

Tariq Mahmood
tmsherazi@ue.edu.pk

Zareen Sakhawat
zareen.s@hotmail.com

Muhammad Saqlain Aslam
saqlain@g.ncu.edu.tw

1 Faculty of Information Technology, Beijing University of Technology, 100124, Beijing, China

2 Department of Information Technology, University of Sialkot, Punjab, 51040, Pakistan

3 Division of Science and Technology, University of Education, Lahore, 54000, Pakistan

4 Department of Computer Science and Information Engineering, National Central University, Taoyuan, 32001, Taiwan