Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 9965426, 16 pages
https://doi.org/10.1155/2022/9965426

Research Article

Improved Arabic Alphabet Characters Classification Using Convolutional Neural Networks (CNN)

Nesrine Wagaa,[1,2,3] **Hichem Kallel**,[1,2,3] **and Nédra Mellouli**[1,2,3]

[1]*National Institute of Applied Sciences and Technology (INSAT) at University of Carthage, LARATSI Laboratory, Cedex 1080, Tunis, Tunisia*
[2]*MedTech at South Mediterranean University, Cedex 1053, Tunis, Tunisia*
[3]*Laboratory of Advanced Computer Science at Paris 8 University, LIASD (EA4383), France*

Correspondence should be addressed to Nesrine Wagaa; nesrinewagah@gmail.com

Received 19 March 2021; Revised 4 November 2021; Accepted 16 December 2021; Published 11 January 2022

Academic Editor: Paolo Gastaldo

Handwritten characters recognition is a challenging research topic. A lot of works have been present to recognize letters of different languages. The availability of Arabic handwritten characters databases is limited. Motivated by this topic of research, we propose a convolution neural network for the classification of Arabic handwritten letters. Also, seven optimization algorithms are performed, and the best algorithm is reported. Faced with few available Arabic handwritten datasets, various data augmentation techniques are implemented to improve the robustness needed for the convolution neural network model. The proposed model is improved by using the dropout regularization method to avoid data overfitting problems. Moreover, suitable change is presented in the choice of optimization algorithms and data augmentation approaches to achieve a good performance. The model has been trained on two Arabic handwritten characters datasets AHCD and Hijja. The proposed algorithm achieved high recognition accuracy of 98.48% and 91.24% on AHCD and Hijja, respectively, outperforming other state-of-the-art models.

1. Introduction

Approximately a quarter of a billion people around the world speak and write the Arabic language [1]. There are a lot of historical books and documents that represent a crucial data set for most Arabic countries written in the Arabic language [1, 2].

Recently, the area of Arabic handwritten characters recognition (AHCR) has received increased research attention [3–5]. It is a challenging topic of computer vision and pattern recognition [1]. This is due to the following:

(i) The difference between handwriting patterns [3].

(ii) The form similarity between Arabic alphabets [1, 3].

(iii) The diacritics of Arabic characters [6].

(iv) As shown in Figure 1, in the Arabic language the shape of each handwritten character depends on its position in the world. For example, here in the word "أماراع" the character "Alif" is written in two different forms "أ" and "ا", where, in the Arabic language, each character has between two and four shapes. Table 1 shows the different shapes of the twenty-eight Arabic alphabets.

With the development of deep learning (DL), convolution neural networks (CNNs) have shown a significant capability to recognize handwritten characters of different languages [3, 7, 8]: Latin [9, 10], Chine [11], Devanagari [12], Malayalam [11], etc.

Most researchers improved the CNN architecture to achieve good handwritten characters recognition performance [6, 13]. However, a neural network with excellent performance usually requires a good tuning of CNN hyperparameters and a good choice of applied optimization algorithms [14–16]. Also, a large amount of training dataset [17, 18] is required to achieve outstanding performance.

The main contributions of this research can be summarized as follows:

Figure 1: One possible connected component box for the word “أمراء”.

Table 1: Twenty-eight shapes of Arabic alphabets.

No.	Name	Isolated	Beginning	Middle	End
1	Alif	ا	ا	إـ	أـ
2	Baa	ب	ـب	ـبـ	بـ
3	Tea	ت	ـت	ـتـ	تـ
4	Thea	ث	ـث	ـثـ	ثـ
5	Jam	ج	-ج	-ج-	ج-
6	Haa	ح	-ح	-ح-	ح-
7	Khaa	خ	-خ	-خ-	خ-
8	Daal	د	د	دـ	دـ
9	Thaal	ذ	ذ	ذـ	ذـ
10	Raa	ر	ر	ر-	ر-
11	Zaay	ز	ز	ز-	ز-
12	Seen	س	ـس	-ـس	س-
13	Sheen	ش	ـش	-ـش	ش-
14	Sad	ص	ـص	-ـص	ص-
15	Dhad	ض	ـض	-ـض	ض-
16	Tah	ط	ـط	ـطـ	طـ
17	Dha	ظ	ـظ	ـظـ	ظـ
18	Ain	ع	-ع	-ع-	ع-
19	Ghen	غ	-غ	-غ-	غ-
20	Fa	ف	ـف	ـفـ	فـ
21	Qaf	ق	-ق	-ق-	ق-
22	Kaf	ك	ـك	ـكـ	كـ
23	Lam	ل	-ل	-ل-	ل-
24	Meem	م	ـم	-ـم	م-
25	Noon	ن	-ن	-ن-	ن-
26	Ha	ه	ـه	ـهـ	هـ
27	Waw	و	و	و-	و-
28	Yaa	ي	-ي	-ي-	ي-

(i) Suggesting a CNN model for recognizing Arabic handwritten characters.

(ii) Tuning of different hyperparameters to improve the model performance.

(iii) Applying different optimization algorithms. Reporting the effectiveness of the best ones.

(iv) Presenting different data augmentation techniques. Reporting the influence of each method on the improvement of Arabic handwritten characters recognition.

(v) Mixing two different Arabic handwritten characters datasets for shape varying. Testing the impact of the presented data augmentation approaches on the mixed dataset.

The rest of this paper is organized as follows. In Section 2, we expose the related works in Arabic handwritten character classification. In Sections 3 and 4, we describe the convolution neural network architecture and the model tuning hyperparameters. In Section 5, we make a detailed description of various used optimization algorithms. In Section 6, we describe the different utilized data augmentation techniques chosen in this study. In Section 7, we provide an overview of the experimental results showing the CNN distinguished performance. Section 8 is conclusion and possible future research directions.

2. Related Work

In recent years, many studies have addressed the classification and recognition of letters, including Arabic handwritten characters. On the other hand, there are a smaller number of proposed approaches for recognizing individual characters in the Arabic language. As a result, Arabic handwritten character recognition is less common compared to English, French, Chinese, Devanagari, Hangul, Malayalam, etc.

Impressive results were achieved in the classification of handwritten characters from different languages, using deep learning models and in particular the CNN.

El-Sawy et al. [6] gathered their own Arabic Handwritten Character dataset (AHCD) from 60 participants. AHCD consists of 16.800 characters. They have achieved a classification accuracy of 88% by using a CNN model consisting of 2 convolutional layers. To improve the CNN performance, regularization and different optimization techniques have been implemented to the model. The testing accuracy was improved to 94.93%.

Altwaijry and Turaiki [13] presented a new Arabic handwritten letters dataset (named “Hijja”). It comprised 47.434 characters written by 591 participants. Their proposed CNN model was able to achieve 88% and 97% testing accuracy, using the Hijja and AHCD datasets, respectively.

Younis [19] designed a CNN model to recognize Arabic handwritten characters. The CNN consisted of three convolutional layers followed by one final fully connected layer. The model achieved an accuracy of 94.7% for the AHCD database and 94.8% for the AIA9K (Arabic alphabet’s dataset).

Latif et al. [20] designed a CNN to recognize a mix of handwriting of multiple languages: Persian, Devanagari, Eastern Arabic, Urdu, and Western Arabic. The input image is of size (28×28) pixels, followed by two convolutional layers, and then a max-pooling operation is applied to both convolution layers. The overall accuracy of the combined multilanguage database was 99.26%. The average accuracy is around 99% for each individual language.

Alrobah and Albahl [21] analyzed the Hijja dataset and found irregularities, such as some distorted letters, blurred symbols, and some blurry characters. They used the CNN model to extract the important features and SVM model for data classification. They achieved a testing accuracy of 96.3%.

Mudhsh et al. [22] designed the VGG net architecture for recognizing Arabic handwritten characters and digits. The model consists of 13 convolutional layers, 2 max-pooling layers, and 3 fully connected layers. Data augmentation and dropout methods were used to avoid the overfitting problem. The model was trained and evaluated by using two different datasets: the ADBase for the Arabic handwritten digits classification topic and HACDB for the Arabic handwritten characters classification task. The model achieved an accuracy of 99.66% and 97.32% for ADBase and HACDB, respectively.

Boufenar et al. [23] used the popular CNN architecture Alexnet. It consists of 5 convolutional layers, 3 max-pooling layers, and 3 fully connected layers. Experiments were conducted on two different databases, OIHACDB-40 and AHCD. Based on the good tuning of the CNN hyperparameters and by using dropout and minibatch techniques, a CNN accuracy of 100% and 99.98% for OIHACDB-40 and AHCD was achieved.

Mustapha et al. [24] proposed a Conditional Deep Convolutional Generative Adversarial Network (CDCGAN) for a guided generation of isolated handwritten Arabic characters. The CDCGAN was trained on the AHCD dataset. They achieved a 10% performance gap between real and generated handwritten Arabic characters.

Table 2 summarizes the literature reviewed for recognizing Arabic handwriting characters using the CNN models. From the previous literature, we notice that most CNN architectures have been trained by using adult Arabic handwriting letters "AHCD". In addition, we observe that most researchers try to improve the performance through the good tuning of the CNN model hyperparameters.

3. The Proposed Arabic Handwritten Characters Recognition System

As shown in Figure 2, the model that we proposed in this study is composed of three principal components: CNN proposed architecture, optimization algorithms, and data augmentation techniques.

In this paper, the proposed CNN model contains four convolution layers, two max-pooling operations, and an ANN model with three fully hidden layers used for the classification. To avoid the overfitting problems and improve the model performance, various optimization techniques were used such as dropout, minipatch, choice of the activation function, etc.

Figure 3 describes the proposed CNN model. Also, in this work, the recognition performance of Arabic handwritten letters was improved through the good choice of the optimization algorithm and by using different data augmentation techniques "geometric transformations, feature space augmentation, noise injection, and mixing images."

4. Convolution Neural Network Architecture

A CNN model [25–34] is a series of convolution layers followed by fully connected layers. Convolution layers allow the extraction of important features from the input data. Fully connected layers are used for the classification of data. The CNN input is the image to be classified; the output corresponds to the predicted class of the Arabic handwritten character.

4.1. Input Data. The input data is an image I of size $(m \times m \times s)$. $(m \times m)$ Defines the width and the height of the image and s denotes the space or number of channels. The value of s is 1 for a grayscale image and equals 3 for a RGB color image.

4.2. Convolution Layer. The convolution layer consists of a convolution operation followed by a pooling operation.

4.2.1. Convolution Operation. The basic concept of the classical convolution operation between an input image I of dimension $(m \times m)$ and a filter F of size $(n \times n)$ is defined as follows (see Figure 4):

$$C = I \otimes F. \tag{1}$$

Here, $\otimes$ denotes the convolution operation. C is the convolution map of size $(a \times a)$, where $a = (m - n + 2p/sL) + 1$. sL is the stride and denotes the number of pixels by which F *is* sliding over I. p is the padding; often it is necessary to add a bounding of zeros around I to preserve complete image information. Figure 4 is an example of the convolution operation between an input image of dimension (8×8) and a filter F of size (3×3). Here, the convolution map C is of size (6×6) with a stride $sL = 1$ and a padding $p = 0$.

Generally, a nonlinear activation function is applied on the convolution map C. The commonly used activation functions are Sigmoid [34–36], Hyperbolic Tangent "Tanh" [35, 37], and Rectified Linear Unit "ReLU" [37, 38] where

$$C_a = f(C), \tag{2}$$

here, C_a is the convolution map after applying the nonlinear activation function f. Figure 5 shows the C_a map when the ReLU activation function is applied on C.

4.2.2. Pooling Operation. The pooling operation is used to reduce the dimension of C_a thus reducing the computational complexity of the network. During the pooling operation, a kernel K of size $(s_p \times s_p)$ is sliding over C_a. s_p denotes the number of patches by which K is sliding over C_a. In our analysis s_p is set to 2. The pooling operation is expressed as

$$P = \text{pool}(C_a), \tag{3}$$

where P is the pooling map and pool is the pooling operation. The commonly used pooling operations are average-pooling, max-pooling, and min-pooling. Figure 6 describes the concept of average-pooling and max-pooling operations using a kernel of size (2×2) and a stride of 2.

4.3. Concatenation Operation. The concatenation operation maps the set of the convoluted images into a vector called the concatenation vector Y.

$$Y = \begin{bmatrix} P_1^c \\ P_2^c \\ \vdots \\ P_n^c \end{bmatrix}, \tag{4}$$

TABLE 2: Summary of Arabic handwritten characters recognition using CNN model.

References	Year	Dataset	Type (size)	Method	Optimization	Accuracy (%)	Loss (%)
El-Sawy et al. [6]	2017	AHCD	Chars (16,800)	CNN	(i) Minibatch	94.93	5.1
Mudhsh et al. [22]	2017	ADBase	Digits (6.600)	CNN (based on VGG net)	(ii) Dropout	99.6	—
		HACDB	Chars (70.000)		(iii) Data augmentation	97.32	—
Boufenar et al. [23]	2017	OIHACDB AHCD	Chars (6.600)	CNN (based on Alexnet)	(i) Dropout (ii) Minibatch	100 99.98	—
Younis [19]	2018	AHCD AIA9K	Chars (8.737)	CNN	—	97.7 94.8	— —
Latif et al. [20]	2018	Mix of handwriting of multiple languages	Chars	CNN	—	99.26	0.02
Altwaijry and Turaiki [13]	2020	Hijja AHCD	Chars (47,434)	CNN	—	88 97	— —
Alrobah &Albahl [21]	2021	Hijja	Chars (47,434)	CNN + SVM	—	96.3	—
Mustapha et al. [24]	2021	AHCD		CDCGAN	—	—	—

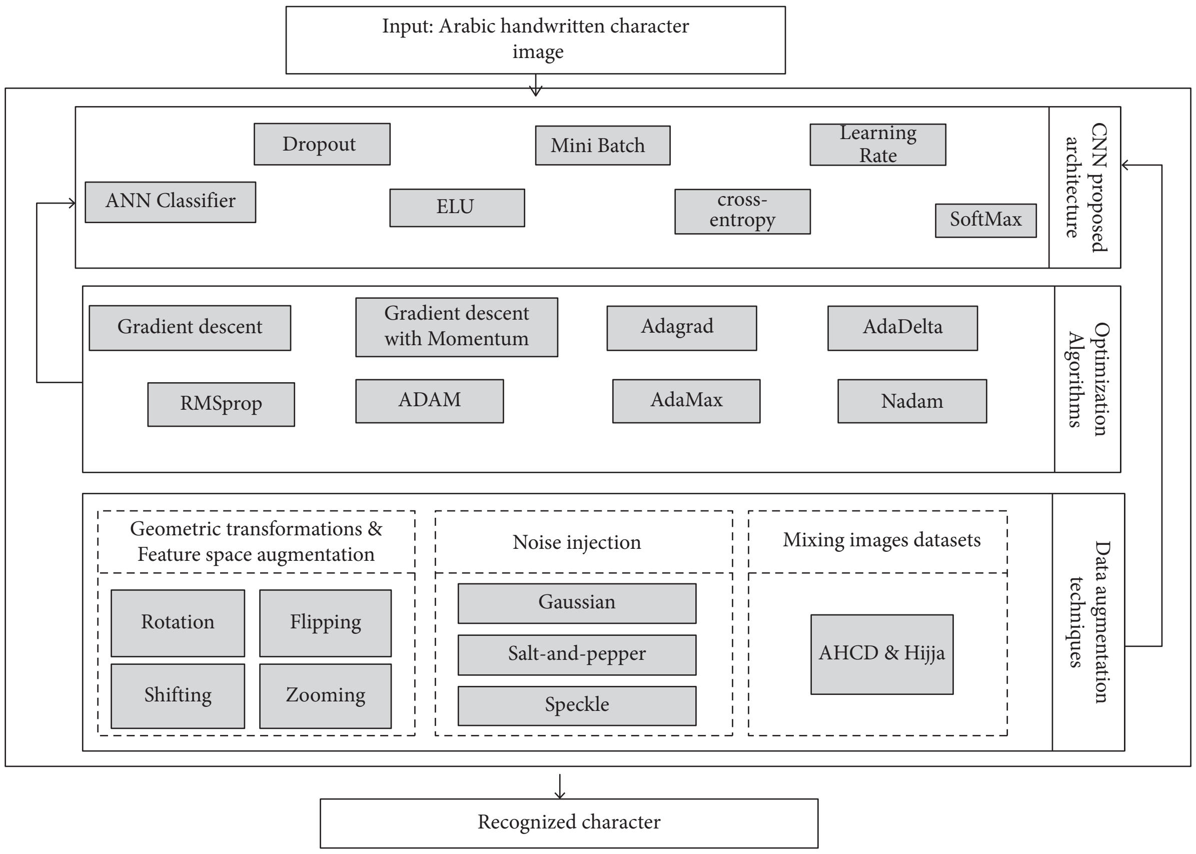


FIGURE 2: Proposed recognition schema for Arabic handwritten characters datasets.

here, P_i^c is the output of the i^{th} convolution layer. n denotes the number of filters applied on the convoluted images P_{i-1}^{c-1}.

4.4. Fully Connected Layer. The CNN classification operation is performed through the fully connected layer [39]. Its input is the concatenation vector Y; the predicted class y is the output of the CNN classifier. The classification operation is performed through a series of t fully connected hidden layers. Each fully connected hidden layer is a parallel collection of artificial neurons. Like synapses in the biological brain, the artificial neurons are connected through weights W. The model output of the i^{th} fully connected hidden layer is

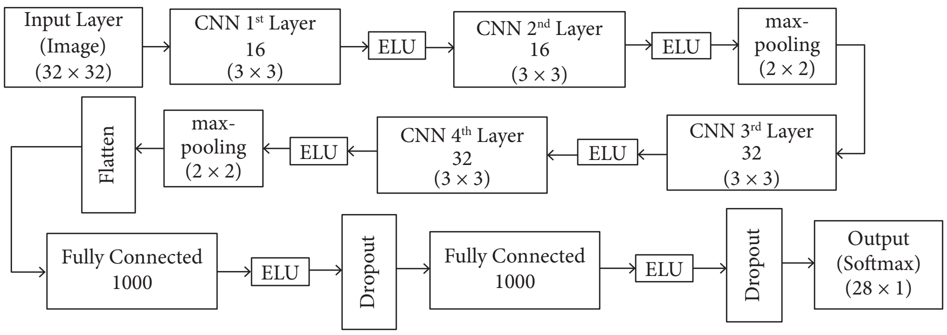


Figure 3: Proposed CNN architecture.

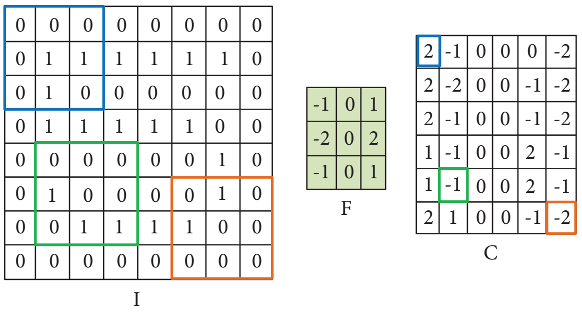


Figure 4: Example of 2D convolution operation [13, 31].

$$Y^i = f(H^i), \quad (5)$$

where the weight sum vector H^i is

$$H^i = W^i Y^{(i-1)} + B^i, \quad (6)$$

here, f is a nonlinear activation function (sigmoid, Tanh, ReLU, etc.). The bias value B^i defines the activation level of the artificial neurons.

5. CNN Learning Process

A trained CNN is a system capable of determining the exact class of a given input data. The training is achieved through an update of the layer's parameters (filters, weights, and biases) based on the error between the CNN predicted class and the class label. The CNN learning process is an iterative process based on the feedforward propagation and back-propagation operations.

5.1. Feedforward Propagation. For the CNN model, the feedforward equations can be derived from (1)–(5) and (6). The Softmax activation [40, 41] function is applied in the final layer to generate the predicted value of the class of the input image I. For a multiclass model, the Softmax is expressed as follows:

$$y_i = \frac{\exp(h_i)}{\sum_j^c \exp(h_j)}, \quad (7)$$

where c denotes the number of classes, y_i is the i^{th} coordinate of the output vector y, and the artificial neural output $h_i = \sum_{j=1}^{n} h_i w_{ij}$.

5.2. Backpropagation. To update the CNN parameters and perform the learning process, a backpropagation optimization algorithm is developed to minimize a selected cost function E. In this analysis, the cross-entropy (CE) cost function [40] is used.

$$E = -\sum_{j=1}^{p} (\check{y}_i \log(y_i) + (1 + \check{y}_i)\log(1 - y_i)), \quad (8)$$

here, $\check{y}_i$ is the desired output (data label).

The most used optimization algorithm to solve classification problems is the gradient descent (GD). Various optimizers for the GD algorithm such as momentum, AdaGrad, RMSprop, Adam, AdaMax, and Nadam were used to improve the CNN performance.

5.2.1. Gradient Descent [40, 42]. GD is the simplest form of optimization gradient descent algorithms. It is easy to implement and gives significant classification accuracy. The general update equation of the CNN parameters using the GD algorithm is

$$\varphi(t+1) = \varphi(t) - \alpha \frac{\partial E}{\partial \varphi(t)}, \quad (9)$$

where φ represents the update of the filters F, the weights W, and the biases B. $(\partial E/\partial \varphi)$ is the gradient with respect to the parameter φ. α is the model learning rate. A too-large value of α may lead to the divergence of the GD algorithm and may cause the oscillation of the model performance. A too-small α stops the learning process.

5.2.2. Gradient Descent with Momentum [43]. The momentum hyperparameter m defines the velocity by which the learning rate α must be increased when the model approaches to the minimal of the cost function E. The update equations using the momentum GD algorithm are expressed as follows:

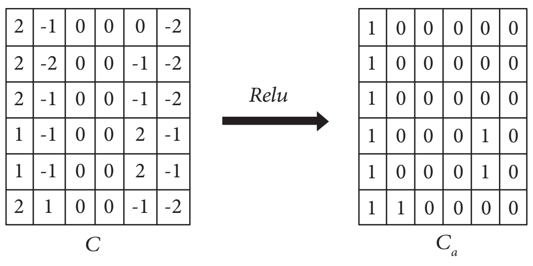


FIGURE 5: Convolution map after applying ReLU activation function [13, 25, 31].

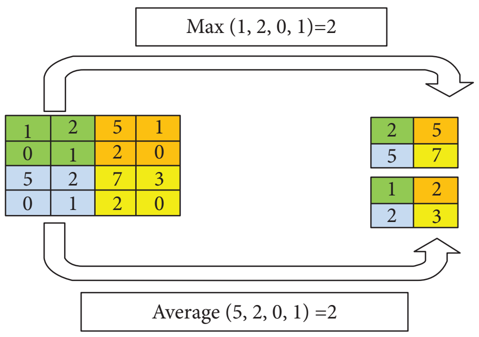


FIGURE 6: Average-pooling, and max-pooling with filter (2×2) and stride 2 [13, 31].

$$v(t) = m\,v(t-1) + \frac{\partial E}{\partial \varphi},$$
$$\varphi(t+1) = \varphi(t) - \alpha\, v(t), \tag{10}$$

where $v(t)$ is the moment gained at t^{th} iteration.

5.2.3. AdaGrad [44]. In this algorithm, the learning rate is a function of the gradient $(\partial E/\partial \varphi)$. It is defined as follows:

$$\alpha(t) = \frac{\beta}{\sqrt{G(t) + \varepsilon}}, \tag{11}$$

where

$$\beta = \frac{\alpha(0)}{\sqrt{\varepsilon}},$$
$$G(t) = \sum_{i=1}^{t} \left(\frac{\partial E}{\partial \varphi(t)} \right)^2, \tag{12}$$

where ϵ is a small smoothing value used to avoid the division by 0 and $G(t)$ is the sum of the squares of the gradients $(\partial E/\partial \varphi(t))$.

With a small magnitude of $(\partial E/\partial \varphi)$, the value of α is increasing. If $(\partial E/\partial \varphi)$ is very large, the value of α is a constant. AdaGrad optimization algorithm changes the learning rate for each parameter at a given time t with considering the previous gradient update. The parameter update equation using AdaGrad is expressed as follows:

$$\varphi(t+1) = \varphi(t) - \alpha(t) \frac{\partial E}{\partial \varphi(t)}. \tag{13}$$

5.2.4. AdaDelta [45]. The issue of AdaGrad is that with much iteration the learning rate becomes very small which leads to a slow convergence. To fix this problem, AdaDelta algorithm proposed to take an exponentially decaying average as a solution, where

$$E\left[G^2(t)\right] = \gamma E\left[G^2(t-1)\right] + (1-\gamma) G^2(t),$$
$$\Delta\theta(t) = \frac{1}{\sqrt{E\left[G^2(t)\right] + \varepsilon}} G(t), \tag{14}$$
$$\varphi(t+1) = \varphi(t) - \alpha \Delta\theta(t),$$

where $E[G^2(t)]$ is the decaying average over past squared gradients and γ is a set usually around 0.9.

5.2.5. RMSprop [45, 46]. In reality, RMSprop is identical to AdaDelta's initial update vector, which we derived above:

$$E\left[G^2(t)\right] = 0.9\, E\left[G^2(t-1)\right] + 0.1 G^2(t),$$
$$\varphi(t+1) = \varphi(t) - \alpha \Delta\theta(t). \tag{15}$$

5.2.6. ADAM [17, 45, 46]. This gradient descent optimizer algorithm computes the learning rate α based on two vectors:

$$r(t) = \beta_1 r(t-1) + (1-\beta_1) \frac{\partial E}{\partial \varphi(t)},$$
$$v(t) = \beta_2 v(t-1) + (1-\beta_2) \left(\frac{\partial E}{\partial \varphi(t)} \right)^2, \tag{16}$$

where $r(t)$ and $v(t)$ are the 1st and the 2nd order moments vectors. β_1 and β_2 are the decay rates. $r(t-1)$ and $v(t-1)$ represent the mean and the variance of the previous gradient.

When $r(t)$ and $v(t)$ are very small, a large step size is needed for parameters update. To avoid this issue, a bias correction value is added to $r(t)$ and $v(t)$.

$$\hat{r}(t) = \frac{r(t)}{\left(1-\beta_1^t\right)},$$
$$\hat{v}(t) = \frac{v(t)}{\left(1-\beta_2^t\right)}, \tag{17}$$

where β_1^t is β_1 power t and β_2^t is β_2 power t.

The Adam update equation is expressed as follows:

$$\varphi(t+1) = \varphi(t) - \alpha \frac{\hat{r}(t)}{\sqrt{\hat{v}(t)} + \varepsilon}. \tag{18}$$

5.2.7. AdaMax [45, 47]. The factor $v(t)$ in the Adam algorithm adjusts the gradient inversely proportionate to the $\ell 2$ norm of previous gradients (via the $v(t-1)$) and current gradient t $(\partial E/\partial\varphi(t))$:

$$v(t) = \beta_2 v(t-1) + (1-\beta_2)\left|\frac{\partial E}{\partial\varphi(t)}\right|^2. \tag{19}$$

The generalization of this update to the ℓp norm is as follows:

$$v(t) = \beta_2^p v(t-1) + \left(1-\beta_2^p\right)\left|\frac{\partial E}{\partial\varphi(t)}\right|^p. \tag{20}$$

To avoid being numerically unstable, ℓ1 and ℓ2 norms are most common in practice. However, in general ℓ∞ also shows stable behavior. As a result, the authors propose AdaMax and demonstrate that $v(t)$ with ℓ∞ converges to the more stable value. Here,

$$\begin{aligned} u(t) &= \beta_2^\infty v(t-1) + (1-\beta_2^\infty)\left|\frac{\partial E}{\partial\varphi(t)}\right|^\infty \\ &= \max\left(\beta_2. v(t-1), \left|\frac{\partial E}{\partial\varphi(t)}\right|\right), \\ \varphi(t+1) &= \varphi(t) - \alpha\frac{\hat{r}(t)}{u(t)}. \end{aligned} \tag{21}$$

5.2.8. Nadam [43]. It is a combination of Adam and NAG, where the parameters update equation using NAG is defined as follows:

$$\begin{aligned} v(t) &= \gamma v(t-1) + \alpha\left(\frac{\partial E}{\partial\varphi(t)} - \gamma v(t-1)\right), \\ \varphi(t+1) &= \varphi(t) - \alpha v(t). \end{aligned} \tag{22}$$

The update equation using Nadam is expressed as follows:

$$\begin{aligned} r &= \left(\beta_1\hat{r}(t) + \frac{(1-\beta_1)}{1-\beta(t)_1}\frac{\partial E}{\partial\varphi(t)}\right), \\ \varphi(t+1) &= \varphi(t) - \frac{\alpha}{\sqrt{\hat{v}(t)+\varepsilon}}r. \end{aligned} \tag{23}$$

6. Data Augmentation Techniques

Deep convolutional neural networks are heavily reliant on big data to achieve excellent performance and avoid the overfitting problem.

To solve the problem of insufficient data for Arabic handwritten characters, we present some basic data augmentation techniques that enhance the size and quality of training datasets.

The image augmentation approaches used in this study include geometric transformations, feature space augmentation, noise injection, and mixing images.

Data augmentation based on geometric transformations and feature space augmentation [17, 48] is often related to the application of rotation, flipping, shifting, and zooming.

6.1. Rotation. The input data is rotated right or left on an axis between 1° and 359°. The rotation degree parameter has a significant impact on the safety of the dataset. For example, on digit identification tasks like MNIST, slight rotations like 1 to 20 or −1 to −20 could be useful, but when the rotation degree increases, properly the CNN network cannot accurately distinguish between some digits.

6.2. Flipping. The input image is flipped horizontally or vertically. This augmentation is one of the simplest to implement and has proven useful on some datasets such as ImageNet and CIFAR-10.

6.3. Shifting. The input image is shifting right, left, up, or down. This transformation is a highly effective adjustment to prevent positional bias. Figure 7 shows an example of shifting data augmentation technique using Arabic alphabet characters.

6.4. Zooming. The input image is zooming, either by adding some pixels around the image or by applying random zooms to the image. The amount of zooming has an influence on the quality of the image; for example, if we apply a lot of zooming, we can lose some image pixels.

6.5. Noise Injection. As it could be seen on Arabic handwritten characters, natural noises are presented in images. Noises make recognition more difficult and for this reason, noises are reduced by image preprocessing techniques. The cos of noise reduction is to perform a high classification, but it causes the alteration of the character shape. The main datasets in this research topic are considered with denoising images. The question which we answer here is how the method could be robust to any noise.

Adding noise [48, 49] to a convolution neural network during training helps the model learn more robust features, resulting in better performance and faster learning. We can add several types of noise when recognizing images, such as the following.

(i) Gaussian noise: injecting a matrix of random values drawn from a Gaussian distribution

(ii) Salt-and-pepper noise: changing randomly a certain amount of the pixels to completely white or completely black

(iii) Speckle noise: only adding black pixels "pepper" or white pixels "salt"

Adding noise to the input data is the most commonly used approach, but during training, we can add random noise to other parts of the CNN model. Some examples include the following:

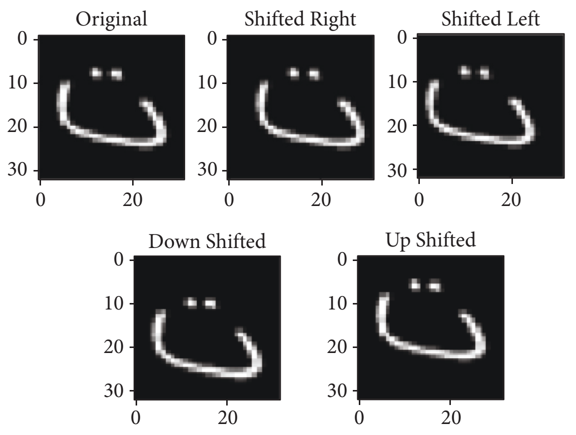


Figure 7: Data augmentation through shifting a single data in four directions by two pixels.

(i) Adding noise to the outputs of each layer
(ii) Adding noise to the gradients to update the model parameters
(iii) Adding noise to the target variables

6.6. Mixing Image's Databases. In this study, we augment the training dataset by mixing two different Arabic handwritten characters datasets, AHCD and Hijja, respectively. AHCD is a clean database, but Hijja is a dataset with very low-resolution images. It comprises many distorted alphabets images.

Then, we evaluate the influence of different mentioned data augmentation techniques (geometric transformations, feature space augmentation, and noise injection) on the recognition performance of the new mixing dataset.

7. Experimental Results and Discussion

7.1. Datasets. In this study, two datasets of Arabic handwritten characters were used: Arabic handwritten characters dataset "AHCD" and Hijja dataset.

AHCD [6] comprises 16.800 handwritten characters of size $(32 \times 32 \times 1)$ pixels. It was written by 60 participants between the ages of 19 and 40 years and most of the participants are right handed. Each participant wrote the Arabic alphabet from "alef" to "yeh" 10 times. The dataset has 28 classes. It is divided into a training set of 13.440 characters and a testing set of 3.360 characters.

Hijja dataset [13] consists of 4.434 Arabic characters of size $(32 \times 32 \times 1)$ pixels. It was written by 591 school children ranging in age between 7 to 12 years. Collecting data from children is a very hard task. Malformed characters are characteristic of children's handwriting; therefore the dataset comprises repeated letters, missing letters, and many distorted or unclear characters. The dataset has 29 classes. It is divided into a training set of 37.933 characters and a testing set of 9.501 characters (80% for training and 20% for test).

Figure 8 shows a sample of AHCD and Hijja Arabic handwritten letters datasets.

7.2. Experimental Environment and Performance Evaluation. In this study the implementation and the evaluation of the CNN model are done out in Keras deep learning environment with TensorFlow backend on Google Colab using GPU accelerator.

We evaluate the performance of our proposed model via the following measures:

Accuracy (A) is a measure for how many correct predictions your model made for the complete test dataset:

$$A = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}. \tag{24}$$

Recall (R) is the fraction of images that are correctly classified over the total number of images that belong to class:

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{25}$$

Precision (P) is the fraction of images that are correctly classified over the total number of images classified:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{26}$$

$F1$ measure is a combination of Recall and Precision measures:

$$F1 = 2 * \frac{P * R}{P + R}. \tag{27}$$

Here, TP = true positive (is the total number of images that can be correctly labeled as belonging to a class x), FP = false positive (represents the total number of images that have been incorrectly labeled as belonging to a class x), FN = false negative (represents the total number of images that have been incorrectly labeled as not belonging to a class x), *TN* = true negative (represents the total number of images that have been correctly labeled as not belonging to a class x).

Also we draw the area under the ROC curve (AUC), where we have the following.

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of all classification thresholds. This curve plots two parameters:

(i) True-positive rate
(ii) False-positive rate

AUC stands for "area under the ROC curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve from (0.0) to (1.1).

7.3. Tuning of CNN Hyperparameters. The objective is to choose the best model that fits the AHCD and Hijja datasets well. Many try-and-error trials in the network configuration tuning mechanism were performed.

The best performance was achieved when the CNN model was constructed of four convolution layers followed by three fully connected hidden layers. The model starts with

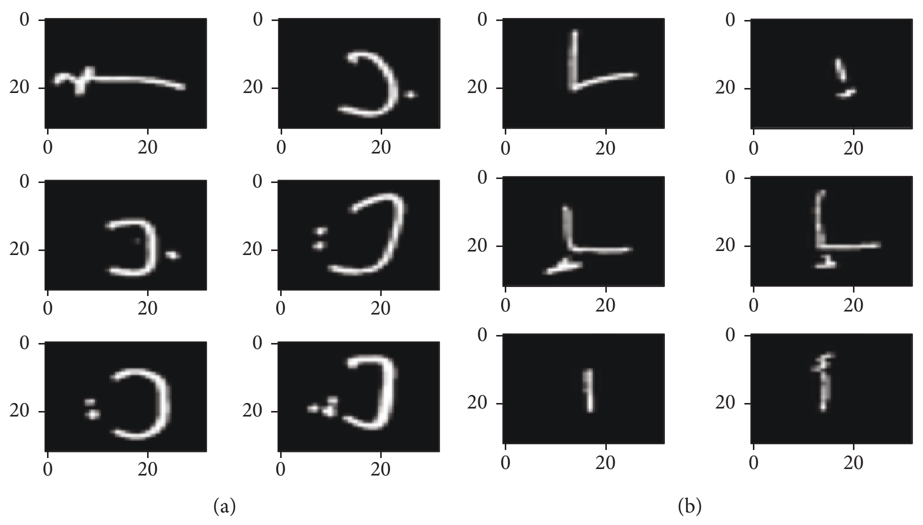


Figure 8: Arabic handwritten letters dataset. (a) AHCD dataset. (b) Hijja dataset.

two convolution layers with 16 filters of size (3×3), then the remaining 2 convolution layers are with 32 filters of size (3×3), and each two convolution layers are followed by max-pooling layers with (2×2) kernel dimension. Finally, three fully connected layers (dense layers) with Softmax activation function to perform prediction. ELU, a nonlinear activation function, was used to remove negative values by converting them into 0.001. The values of weights and bias are updated by a backward propagation process to minimize the loss function.

To reduce the overfitting problem a dropout of 0.6 rate is added to a model between the dense layers and applies to outputs of the prior layer that are fed to the subsequent layer. The optimized parameters used to improve the CNN performance were as follows: Optimizer algorithm is Adam, the loss function is the cross-entropy, learning rate = 0.001, batch size = 16, and epochs = 40.

We compare our model to CNN-for-AHCD over both the Hijja dataset and the AHCD dataset. The code for CNN-for-AHCD is available online [31], which allows comparison of its performance over various datasets.

On the Hijja dataset, which has 29 classes, our model achieved an average overall test set accuracy of 88.46%, precision of 87.98%, recall of 88.46%, and an F1 score of 88.47%, while CNN-for-AHCD achieved an average overall test set accuracy of 80%, precision of 80.79%, recall of 80.47%, and an F1 score of 80.4%.

On the AHCD dataset, which has 28 classes, our model achieved an average overall test set accuracy of 96.66%, precision of 96.75%, recall of 96.67%, and an F1 score of 96.67%, while CNN-for-AHCD achieved an average overall test set accuracy of 93.84%, precision of 93.99%, recall of 93.84%, and an F1 score of 93.84%.

The detailed metrics are reported per character in Table 3.

We note that our model outperforms CNN-for-AHCD by a large margin on all metrics.

Figure 9 shows the testing result AUC of AHCD and Hijja dataset.

7.4. Optimizer Algorithms. The objective is to choose the best optimizers algorithms that fit the AHCD and Hijja best performance. In this context, we tested the influence of the following algorithms on the classification of handwritten Arabic characters:

(i) Adam
(ii) SGD
(iii) RMSprop
(iv) AdaGrad
(v) Nadam
(vi) Momentum
(vii) AdaMax

By using Nadam optimization algorithm, on the Hijja dataset, our model achieved an average overall test set accuracy of 88.57%, precision of 87.86%, recall of 87.98%, and an F1 score of 87.95%.

On the AHCD dataset, our model achieved an average overall test set accuracy of 96.73%, precision of 96.80%, recall of 96.73%, and an F1 score of 96.72%.

The detailed results of different optimizations algorithms are mentioned in Table 4.

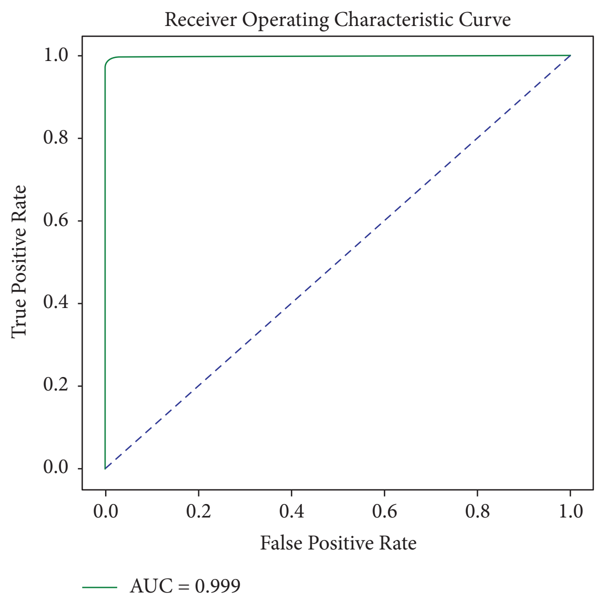
7.5. Results of Data Augmentation Techniques. Generally, the neural network performance is improved through the good tuning of the model hyperparameters. Such improvement in the CNN accuracy is linked to the availability of training dataset. However, the networks are heavily reliant on big data to avoid overfitting problem and perform well.

Data augmentation is the solution to the problem of limited data. The image augmentation techniques used and discussed in this study include geometric transformations and feature space augmentation (rotation, shifting, flipping, and zooming), noise injection, and mixing images from two different datasets.

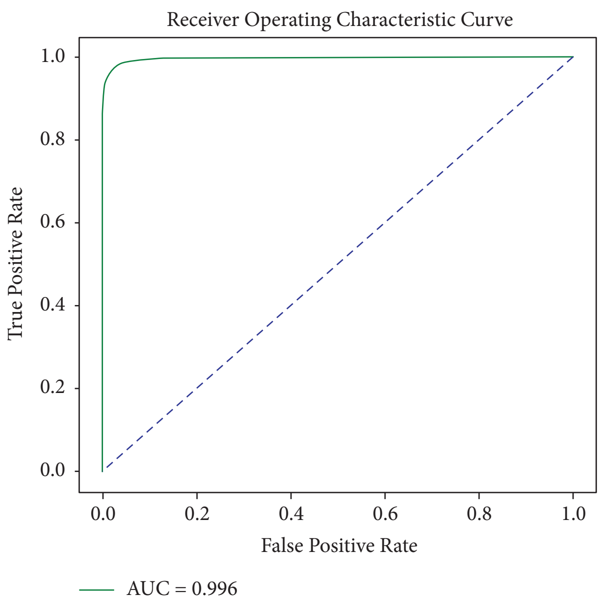
For the geometric transformations and feature space augmentation, we try to well choose the percentage of

Table 3: Experimental results on the Hijja and AHCD datasets via CNN-for-AHCD model.

Characters	Hijja dataset						AHCD dataset					
	CNN-for-AHCD			Our model			CNN-for-AHCD			Our model		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1. Alif	0.93	0.97	0.95	0.97	0.98	0.98	0.96	0.99	0.98	0.96	1.00	0.98
2. Baa	0.82	0.91	0.86	0.93	0.96	0.94	0.97	0.97	0.97	0.99	0.97	0.98
3. Tea	0.66	0.88	0.75	0.84	0.92	0.88	0.87	0.95	0.91	0.96	0.95	0.95
4. Thea	0.76	0.81	0.78	0.94	0.93	0.93	0.95	0.88	0.92	0.97	0.95	0.96
5. Jam	0.79	0.85	0.82	0.80	0.90	0.84	0.95	0.96	0.95	0.99	0.98	0.99
6. Haa	0.83	0.60	0.70	0.85	0.89	0.87	0.93	0.93	0.93	0.99	0.96	0.97
7. Khaa	0.76	0.77	0.77	0.82	0.73	0.77	0.94	0.93	0.93	0.99	0.96	0.97
8. Daal	0.65	0.69	0.67	0.79	0.68	0.73	0.91	0.94	0.93	0.95	0.88	0.92
9. Thaal	0.70	0.68	0.69	0.83	0.92	0.87	0.96	0.91	0.93	0.88	0.95	0.92
10. Raa	0.86	0.87	0.87	0.81	0.91	0.86	0.89	0.98	0.94	0.94	0.99	0.96
11. Zaay	0.87	0.89	0.88	0.96	0.90	0.93	0.94	0.88	0.91	0.96	0.92	0.94
12. Seen	0.84	0.92	0.88	0.95	0.92	0.94	0.95	0.91	0.93	1.00	0.97	0.99
13. Sheen	0.86	0.82	0.84	0.90	0.88	0.89	0.92	0.98	0.95	0.99	1.00	1.00
14. Sad	0.75	0.81	0.78	0.90	0.86	0.88	0.84	0.96	0.90	0.96	0.97	0.97
15. Dhad	0.80	0.76	0.78	0.91	0.93	0.92	1.00	0.89	0.94	0.97	0.95	0.96
16. Tah	0.90	0.83	0.87	0.96	0.89	0.92	0.96	0.94	0.95	0.94	0.98	0.96
17. Dha	0.83	0.87	0.85	0.84	0.82	0.83	0.97	0.94	0.95	0.97	0.95	0.96
18. Ain	0.74	0.70	0.71	0.86	0.86	0.86	0.95	0.90	0.92	0.98	0.98	0.98
19. Ghen	0.83	0.71	0.77	0.75	0.86	0.80	0.89	0.97	0.93	0.98	0.99	0.99
20. Fa	0.77	0.65	0.71	0.92	0.87	0.89	0.92	0.84	0.88	0.94	0.98	0.96
21. Qaf	0.81	0.80	0.81	0.87	0.89	0.88	0.87	0.91	0.89	0.99	0.95	0.97
22. Kaf	0.86	0.78	0.82	0.93	0.89	0.91	0.98	0.96	0.97	0.98	0.93	0.96
23. Lam	0.90	0.87	0.88	0.91	0.89	0.90	0.98	0.97	0.98	0.99	0.98	0.99
24. Meem	0.83	0.85	0.84	0.82	0.80	0.81	0.98	0.98	0.98	0.98	0.98	0.98
25. Noon	0.70	0.77	0.73	0.88	0.83	0.86	0.92	0.92	0.92	0.86	0.97	0.91
26. Ha	0.81	0.76	0.78	0.89	0.94	0.91	0.97	0.96	0.96	0.97	0.96	0.97
27. Waw	0.930	0.82	0.87	0.94	0.92	0.93	0.96	0.94	0.95	0.97	0.94	0.95
28. Yaa	0.82	0.81	0.82	0.87	0.82	0.84	0.97	0.97	0.97	0.99	0.98	0.99
29. Hamza	0.74	0.73	0.74				na	na	na	na	na	na
Acc. (train)			0.88			**0.98**			0.91			**1.00**
Acc. (test)			0.80			**0.88**			0.94			**0.96**
Macro avg	0.81	0.80	0.80	0.88	0.88	0.88	0.94	0.94	0.94	0.97	0.96	0.96
Weighted avg	0.81	0.80	0.80	0.89	0.88	0.88	0.94	0.94	0.94	0.97	0.96	0.96



(a)



(b)

Figure 9: AUC curve of Arabic handwritten characters dataset. (a) AUC of AHCD testing dataset. (b) AUC of Hijja testing dataset.

Table 4: Experimental results on the Hijja and AHCD through different optimizers algorithms.

	Accuracy (%)				Precision (%)				Recall (%)				F1 score (%)			
Algorithm	AHCD		Hijja		AHCD		Hijja		AHCD		Hijja		AHCD		Hijja	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Adam	99.76	96.66	97.60	88.46	99.76	96.75	97.40	87.98	99.76	96.67	97.60	88.46	99.76	96.67	97.60	88.47
SGD	99.44	95.68	94.16	87.85	99.45	95.84	94.28	87.93	99.45	95.68	94.16	87.85	99.45	95.70	94.17	87.88
RMSprop	99.72	96.22	96.55	87.98	99.73	96.30	94.28	87.86	99.72	96.22	94.16	87.98	99.72	96.23	94.17	87.95
AdaGrad	88.62	83.54	63.90	60.37	88.73	83.78	87.86	64.95	88.62	83.54	87.98	63.91	88.60	83.52	87.95	63.77
Nadam	**99.77**	**96.73**	**97.76**	**88.57**	**99.77**	**96.80**	**97.57**	**87.86**	**99.77**	**96.73**	**97.76**	**87.98**	**99.77**	**96.72**	**97.76**	**87.95**
Momentum	99.16	95.56	94.16	87.76	99.18	95.72	94.11	87.73	99.17	95.57	94.17	87.77	99.17	95.58	94.15	87.74
AdaMax	99.89	96.19	97.19	87.57	99.90	96.25	97.01	87.86	99.90	96.19	97.19	87.98	99.90	96.19	97.20	87.95

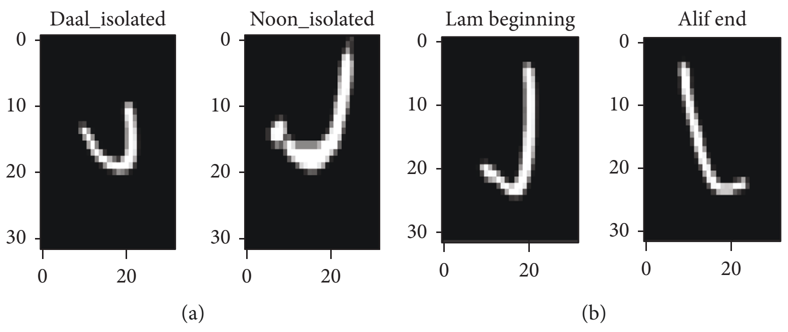


Figure 10: Example of Arabic handwritten characters that cannot be properly distinguished when applying the rotation and flipping data augmentation techniques. (a) Confusion caused by rotation. (b) Confusion caused by flipping.

Table 5: Experimental results of data augmentation techniques on Hijja and AHCD datasets.

Dataset	Data augmentation techniques with using Nadam algorithm									
	Rotation		Shifting		Rotation and shifting		Flipping		Zooming	
	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy
AHCD	99.82	97.64	99.56	98.09	99.41	**98.48**	99.69	97.29	99.85	98.00
Hijja	97.18	89.86	92.72	90.28	91.73	**91.24**	95.39	88.94	99.16	88.66

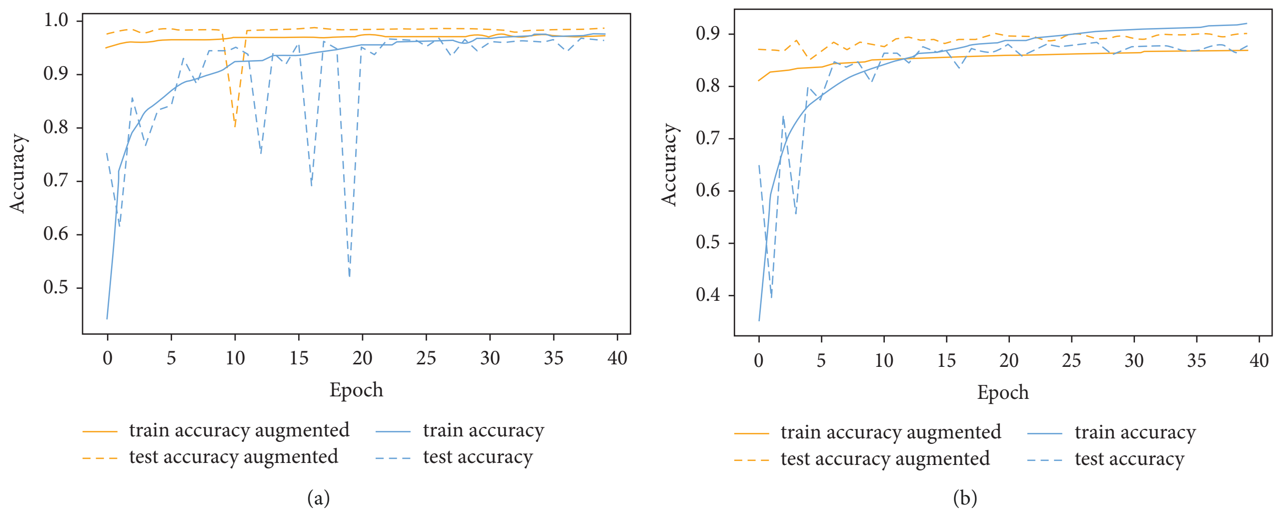


Figure 11: Rotation and shifting data augmentation results of Arabic handwritten characters datasets. (a) Rotation and shifting results of AHCD dataset. (b) Rotation and shifting results of Hijja dataset.

rotation, shifting, flipping, and zooming for the model attending a good performance. For example, if we rotate the Latin handwritten number database (MNIST) by 180°, the network will not be able to accurately distinguish between the handwritten digits “6” and “9”. Likewise, on the AHCD and Hijja datasets, if rotating or flipping techniques are used the network will be unable to distinguish between some handwritten Arabic characters. For example, as shown in Figure 10, with a rotation of 180°, the character Daal isolated (د) will be the same as the character Noon isolated (ن).

The detailed results of rotation, shifting, flipping, and zooming data augmentation techniques are mentioned in Table 5.

As shown in Table 5 and Figure 11, by using rotation and shifting augmentation approaches, our model achieved a testing accuracy of 98.48% and 91.24% on AHCD dataset and Hijja dataset, respectively. We achieved this accuracy through rotating the input image by 10° and shifting it just by one pixel.

Adding noise is a technique used to augment the training input data. Also in most of the cases, this is bound to increase the robustness of our network.

In this work we used the three types of noise to augment our data:

(i) Gaussian noise
(ii) Salt-and-pepper noise

Table 6: Experimental results on the Hijja and AHCD through noise injection.

	Type of noise injection with using Nadam algorithm					
	Gaussian noise		Salt-and-pepper noise		Speckle noise	
	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy
AHCD	99.17	96.78	99.16	97.15	99.14	96.82
Hijja	97.18	89.85	92.96	90.10	91.32	89.73

Table 7: Experimental results on mixing of Hijja and AHCD through different data augmentation techniques.

Mixed dataset		Data augmentation techniques with using Nadam algorithm									
		Rotation		Shifting		Rotation and shifting		Flipping		Gaussian noise	
Training	Testing	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy
80(%): AHCD 20(%): Hijja	20(%): AHCD 10(%): Hijja	99.62	97.42	99.36	98.02	99.38	**98.32**	99.77	97.08	99.15	96.74
80(%): Hijja 20(%): AHCD	20(%): Hijja 10(%): AHCD	97.36	88.47	93.66	89.07	94.91	**90.54**	95.21	88.21	96.68	88.49
80(%): AHCD 80(%): Hijja	20(%): AHCD 20(%): Hijja	97.27	74.53	97.16	75.13	98.13	**78.13**	98.16	74.22	96.98	74.02

(iii) Speckle noise

The detailed results of different types of noise injection are mentioned in Table 6. As shown by adding different types of noise, the model accuracy is improved, which demonstrate the robustness of our proposed architecture. We achieved good results when adding noise to the outputs of each layer.

The proposed idea in this study is to augment the number of training databases by mixing the two datasets AHCD and Hijja, and then we apply the previously mentioned data augmentation methods on the new mixed dataset. Our purpose to use malformed handwritten characters as it proposes the Hijja dataset is to improve the accuracy of our method with noised data.

The detailed results of data augmentation techniques on the mixed database are mentioned in Table 7. As shown, the model performance depends on the rate of using Arabic handwriting "Hijja" database. The children had trouble following the reference paper, which results in very low-resolution images comprising many unclear characters. Therefore mixing the datasets would certainly reduce performance.

8. Conclusions and Possible Future Research Directions

In this paper, we proposed a convolution neural network (CNN) to recognize Arabic handwritten characters dataset. We have trained the model on two Arabic datasets AHCD and Hijja. By the good tuning of the network hyperparameters, we achieved an accuracy of 96.73% and 88.57% on AHCD and Hijja.

To improve the model performance, we have implemented different optimization algorithms. For both databases, we achieved an excellent performance by using Nadam optimizer.

To solve the problem of insufficient Arabic handwritten datasets, we have applied different data augmentation techniques. The augmentation approaches are based on geometric transformation, feature space augmentation, noise injection, and mixing of datasets.

By using rotation and shifting techniques, we achieved a good accuracy equal to 98.48% and 91.24% on AHCD and Hijja.

To improve the robustness of the CNN model and increase the number of training datasets, we added three types of noise (Gaussian noise, Salt-and-pepper, and Speckle noise).

Also in this work we first augmented the database by mixing two Arabic handwritten characters datasets; then we tested the results of the previously mentioned data augmentation techniques on the new mixed dataset, where the first database "AHCD" comprises clear images with a very good resolution, but the second database "Hijja" has many distorted characters. Experimentally show that the geometric transformations (rotation, shifting, and flipping), feature space augmentation, and noise injection always improve the network performance, but the rate of using the unclean database "Hijja" harms the model accuracy.

An interesting future direction is the cleaning and processing of Hijja dataset to eliminate the problem of low-resolution and unclear images and then the implementation of the proposed CNN network and data augmentation techniques on the new mixed and cleaned database.

In addition, we are interested in evaluating the result of other augmentation approaches, like adversarial training, neural style transfer, and generative adversarial networks on the recognition of Arabic handwritten characters dataset. We plan to incorporate our work into an application for children that teaches Arabic spelling.

Abbreviations

AHCR:	Arabic handwritten characters recognition
DL:	Deep learning
CNNs:	Convolution neural networks
AHCD:	Arabic handwritten character dataset
SVM:	Support vector machine
ADBase:	Arabic digits database
HACDB:	Handwritten Arabic characters database
OIHACDB:	Offline handwritten Arabic character database
CDCGAN:	Conditional deep convolutional generative adversarial network
Tanh:	Hyperbolic tangent
ReLU:	Rectified linear unit
CE:	Cross-entropy
GD:	Gradient descent
NAG:	Nesterov accelerated gradient
TP:	True positive
FP:	False positive
FN:	False Negative
TN:	True negative
AUC:	Area under curve
ROC:	Receiver operating curve
ELU:	Exponential linear unit

Symbols

I:	Image
m:	Width and height of the image
c:	Number of channels
F:	Filter
n:	Filter size
$\otimes$:	Convolution operation
C:	Convolution map
a:	Size of convolution map
S_c:	Stride
p:	Padding
f:	Nonlinear activation function
C_a:	Convolution map after applying f
K:	Kernel
S_p:	Number of patches
pool:	Pooling operation
P:	Pooling map
Y^0:	Concatenation vector
P_i^c:	Output of the convolution layer
P_{i-1}^{c-1}:	Convoluted image
Y^{i-1}:	Input of the ith fully connected hidden layer
Y^i:	Output of the ith fully connected hidden layer
H^i:	Weight sum vector
B^i:	Bias
E:	Cost function
$\check{y}_i$:	Desired output
ϕ:	Update of the filter F
$(\partial E/\partial\varphi)$:	Gradient
α:	Model learning
m:	Momentum
$v(t)$:	Moment gained at the ith iteration
ε:	Smoothing value
$G(t)$:	Sum of the squares of the gradient
$E[G(t)^2]$:	Decaying overage
$r(t)$:	Moments vector
β:	Decay rate
$r(t-1)$:	Mean of the previous gradient
$v(t-1)$:	Variance of the previous gradient.

Data Availability

Previously reported AHCD data were used to support this study and are available at https://www.kaggle.com/mloey1/ahcd1. These prior studies (and datasets) are cited at relevant places within the text as [43].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

References

[1] Y. M. Alginahi, "Arabic character segmentation: a survey," *International Journal on Document Analysis and Recognition*, vol. 16, no. 2, pp. 105–126, 2013.

[2] M. T. Parvez and S. A. Mahmoud, "Offline arabic handwritten text recognition: a survey," *ACM Computing Surveys*, vol. 45, no. 2, pp. 1–35, 2013.

[3] D. K. Sahu and C. V. Jawahar, "Unsupervised feature learning for optical character recognition," in *Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1041–1045, Tunis, Tunisia, August 2015.

[4] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez, "Offline continuous handwriting recognition using sequence to sequence neural networks," *Neurocomputing*, vol. 289, pp. 119–128, 2018.

[5] J. Bai, Z. Chen, B. Feng, and B. Xu, "Image character recognition using deep convolutional neural network learned from different languages," in *Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP)*, pp. 2560–2564, Paris, France, October 2014.

[6] A. El-Sawy, M. Loey, and E. Hazem, "Arabic handwritten characters recognition using convolutional neural network," *WSEAS Transactions on Computer Research*, vol. 5, pp. 11–19, 2017.

[7] J. Xiao, Z. Xuehong, H. Chuangxia, Y. Xiaoguang, W. Fenghua, and M. Zhong, "A new approach for stock price analysis and prediction based on SSA and SVM," *International Journal of Information Technology Decision Making*, vol. 18, pp. 287–310, 2019.

[8] N. Wagaa and H. Kallel, "Vector-based back propagation algorithm of supervised convolution neural network," in *Proceedings of the International Conference on Control,*

Automation and Diagnosis (ICCAD), Paris, France, October 2020.

[9] Y. Liang, J. Wang, S. Zhou, Y. Gong, and N. Zheng, "Incorporating image priors with deep convolutional neural networks for image super-resolution," *Neurocomputing*, vol. 194, pp. 340–347, 2016.

[10] H. M. Najadat, A. A. Alshboul, and A. F. Alabed, "Arabic handwritten characters recognition using convolutional neural network," in *Proceedings of the 10th International Conference on Information and Communication Systems*, Irbid, Jordan, June 2019.

[11] H. Kaur and S. Rani, "Handwritten gurumukhi character recognition using convolution neural network," *International Journal of Computational Intelligence Research*, vol. 13, pp. 933–943, 2017.

[12] P. Ahamed, S. Kundu, T. Khan, V. Bhateja, R. Sarkar, and A. Faruk Mollah, "Handwritten arabic numerals recognition using convolutional neural network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 5445–5457, 2020.

[13] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Computing Applications*, vol. 33, 2020.

[14] K. V. Greeshma and K. Sreekumar, "Hyperparameter optimization and regularization on fashion-MNIST classification," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, pp. 2277–3878, 2019.

[15] O. Y. Bakhteev and V. V. Strijov, "Comprehensive analysis of gradient-based hyperparameter optimization algorithms," *Annals of Operations Research*, vol. 289, 2020.

[16] S. Derya, "A comparison of optimization algorithms for deep learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, pp. 51–65, 2020.

[17] A. H. García and P. König, "Further advantages of data augmentation on convolutional neural networks," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 95–103, Springer, Cham, Switzerland, September 2018.

[18] A. A. Hidayata, K. Purwandaria, T. W. Cenggoroa, and B. Pardamean, "A convolutional neural network-based ancient sundanese character classifier with data augmentation," in *Proceedings of the 5th International Conference on Computer Science and Computational Intelligence*, pp. 195–201, Indonesia, 2021.

[19] K. Younis, "Arabic handwritten characters recognition based on deep convolutional neural networks," *Jordan Journal Computers and Information Technology (JJCIT)*, vol. 3, 2018.

[20] G. Latif, J. Alghazo, L. Alzubaidi, M. M. Naseer, and Y. Alghazo, "Deep convolutional neural network for recognition of unified multi-language handwritten numerals," in *Proceedings of the 2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)*, pp. 90–95, London, UK, March 2018.

[21] N. Alrobah and S. Albahli, "A hybrid deep model for recognizing arabic handwritten characters," *IEEE Acces*, vol. 9, pp. 87058–87069, 2021.

[22] M. A. Mudhsh and R. Almodfer, "Arabic handwritten alphanumeric character recognition using very deep neural network," *MDPI, Information*, vol. 8, 2017.

[23] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten arabic character recognition," *Cognitive Systems Research*, vol. 50, 2017.

[24] I. B. Mustapha, S. Hasan, H. Nabus, and S. M. Shamsuddin, "Conditional deep convolutional generative adversarial networks for isolated handwritten arabic character generation," *Arabian Journal for Science and Engineering*, 2021.

[25] A. Yuan, G. Bai, L. Jiao, and Y. Liu, "Offline handwritten English character recognition based on convolutional neural network," in *Proceedings of the 2012 10th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 125–129, IEEE, Gold Coast, Australia, March 2012.

[26] M. Mhapsekar, P. Mhapsekar, A. Mhatre, and V. Sawant, "Implementation of residual network (ResNet) for devanagari handwritten character recognition," *Algorithms for Intelligent Systems*, pp. 137–148, 2020.

[27] N. K. Pius and A. Johny, "Malayalam handwritten character recognition system using convolutional neural network," *International Journal of Applied Engineering Research*, vol. 15, pp. 918–920, 2020.

[28] R. S. Hussien, A. A. Elkhidir, and M. G. Elnourani, "Optical character recognition of arabic handwritten characters using neural network," in *Proceedings of the 2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, pp. 456–461, IEEE, Khartoum, Sudan, September 2015.

[29] A. ElAdel, R. Ejbali, M. Zaied, and C. B. Amar, "Dyadic multiresolution analysis-based deep learning for arabic handwritten character classification," in *Proceedings of the 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 807–812, IEEE, Vietri sul Mare, Italy, November 2015.

[30] M. Elleuch, N. Tagougui, and M. Kherallah, "Arabic handwritten characters recognition using deep belief neural networks," in *Proceedings of the 2015 12th International Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 1–5, IEEE, Mahdia, Tunisia, March 2015.

[31] A. Elsawy, M. Loey, and H. M El-Bakry, "Arabic handwritten characters recognition using convolutional neural network," *WSEAS Transactions on Computer Research*, vol. 5, pp. 11–19, 2017.

[32] K. M. M. Yaagoup and M. E. M. Mus, "Online Arabic handwriting characters recognition using deep learning," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 9, 2020.

[33] M. Shams, A. Elsonbaty, and W. ElSawy, "Arabic handwritten character recognition based on convolution neural networks and support vector machine," *International Journal of Advanced Computer Science and Applications*, vol. 11, 2020.

[34] S. Narayan, "The generalized sigmoid activation function: competitive supervised learning," *Information Sciences*, vol. 99, no. 1-2, pp. 69–82, 1997.

[35] B. L. Kalman and C. Kwasny, "Why tanh: choosing a sigmoidal function," in *Proceedings of the JCNN International Joint Conference on Neural Networks*, IEEE, Baltimore, MD, USA, June 1992.

[36] B. I. Dlimi and H. Kallel, "Robust Neural Control for Robotic Manipulators," *International journal of Enhanced Research in Science*, Technology and Engineering, IJERSTE, vol. 5, no. 2, pp. 198–205, 2016.

[37] P. Sibi, S. Allwyn Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," *Journal of Theoretical and Applied Information Technology*, vol. 47, 2013.

[38] G. Lin and W. Shen, "Research on convolutional neural network based on improved relu piecewise activation function," *Procedia Computer Science*, vol. 131, pp. 977–984, 2018.

[39] M. Kubat, "Artificial neural networks," *An Introduction to Machine Learning*, pp. 91–111, 2015.

[40] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” *Proceedings of COMPSTAT’2010*, pp. 177–186, 2010.

[41] D. Wang, L. Huang, and L. Tang, “Dissipativity and synchronization of generalized BAM neural networks with multivariate discontinuous activations,” *IEEE Transactions on Neural Network Learning System*, vol. 29, pp. 3815–3827, 2017.

[42] R. Sutton, “Two problems with back propagation and other steepest descent learning procedures for networks,” in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pp. 823–832, Chicago, IL, USA, December 1986.

[43] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the International Conference on Machine Learning*, pp. 1139–1147, Atlanta, GA, USA, June 2013.

[44] J. Duchi, E. Hazan, and Y. Singer, “Adaptive sub gradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[45] S. Derya, “A comparison of optimization algorithms for deep learning,” *International Journal of Pattern Recognition and Artificial Intelligence*, 2020.

[46] G. Hinton, N. Srivastava, and K. Swersky, *rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude*, 2012.

[47] D. P. Kingma and L. J. Ba, “Adam: a method for stochastic optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[48] S. Connor and M. K. Taghi, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, 2019.

[49] P. Panda and K. Roy, “Implicit advers arial data augmentation and robustness with noise-based learning,” *Neural Networks*, vol. 141, pp. 1139–1147, 2021.