Research Article

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

Nesrine Wagaa,1,2,3 Hichem Kallel,1,2,3 and Nédra Mellouli1,2,3

1National Institute of Applied Sciences and Technology (INSAT) at University of Carthage, LARATSI Laboratory, Cedex 1080, Tunis, Tunisia
2MedTech at South Mediterranean University, Cedex 1053, Tunis, Tunisia
3Laboratory 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

Copyright © 2022 Nesrine Wagaa et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 “تارمأ” the character “Alif” is written in two different forms “أ” and “إ”, 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:
(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%.

Altwajry 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 \((28 \times 28)\) 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%.

Mudsh 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
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):

$$C = I \otimes F.$$  

Here, $\otimes$ denotes the convolution operation. $C$ is the convolution map of size $(a \times a)$, where $a = (m - n + 2p/s_L) + 1$. $s_L$ 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 $(8 \times 8)$ and a filter $F$ of size $(3 \times 3)$. Here, the convolution map $C$ is of size $(6 \times 6)$ with a stride $s_L = 1$ 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

$$C_a = f(C),$$

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

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

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 $(2 \times 2)$ 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 $Y$.

$$Y = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix}.$$ 

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

### 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 $f$ 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

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

<table>
<thead>
<tr>
<th>References</th>
<th>Year</th>
<th>Dataset</th>
<th>Type (size)</th>
<th>Method</th>
<th>Optimization</th>
<th>Accuracy (%)</th>
<th>Loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>El-Sawy et al. [6]</td>
<td>2017</td>
<td>AHCD</td>
<td>Chars (16,800)</td>
<td>CNN</td>
<td>(i) Minibatch</td>
<td>94.93</td>
<td>5.1</td>
</tr>
<tr>
<td>Mudhsh et al. [22]</td>
<td>2017</td>
<td>ADBase</td>
<td>Digits (6,600)</td>
<td>CNN (based on VGG net)</td>
<td>(ii) Dropout</td>
<td>99.6</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HACDB</td>
<td>Chars (70,000)</td>
<td></td>
<td>(iii) Data</td>
<td>97.32</td>
<td>—</td>
</tr>
<tr>
<td>Bounenar et al. [23]</td>
<td>2017</td>
<td>OHACDB</td>
<td>Chars (6,600)</td>
<td>CNN (based on Alexnet)</td>
<td>(i) Dropout</td>
<td>100</td>
<td>99.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AHCD</td>
<td></td>
<td></td>
<td>(ii) Minibatch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Younis [19]</td>
<td>2018</td>
<td>AHCD</td>
<td>Chars (8,737)</td>
<td>CNN</td>
<td></td>
<td>97.7</td>
<td>94.8</td>
</tr>
<tr>
<td>Latif et al. [20]</td>
<td>2018</td>
<td>Mix of handwriting of multiple languages</td>
<td>Chars</td>
<td>CNN</td>
<td>—</td>
<td>99.26</td>
<td>0.02</td>
</tr>
<tr>
<td>Alrobah &amp; Albahl [21]</td>
<td>2021</td>
<td>Hijja</td>
<td>Chars (47,434)</td>
<td>CNN + SVM</td>
<td>—</td>
<td>96.3</td>
<td>—</td>
</tr>
<tr>
<td>Mustapha et al. [24]</td>
<td>2021</td>
<td>AHCD</td>
<td></td>
<td>CDCGAN</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

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

Input: Arabic handwritten character image

<table>
<thead>
<tr>
<th>Dropout</th>
<th>Mini Batch</th>
<th>Learning Rate</th>
<th>ELU</th>
<th>cross-entropy</th>
<th>SoftMax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN Classifier</td>
<td>Gradient descent</td>
<td>Gradient descent with Momentum</td>
<td>ADAM</td>
<td>AdaMax</td>
<td>AdaDelta</td>
</tr>
<tr>
<td>RMSprop</td>
<td>ADAGRAD</td>
<td>ADADELTA</td>
<td>ADAM</td>
<td>ADAM</td>
<td>ADAM</td>
</tr>
<tr>
<td>Gradient descent</td>
<td>Gradient descent with Momentum</td>
<td>ADAGRAD</td>
<td>ADADELTA</td>
<td>ADADELTA</td>
<td>ADADELTA</td>
</tr>
<tr>
<td>Rotation</td>
<td>Flipping</td>
<td>Shifting</td>
<td>Zooming</td>
<td>Gaussian</td>
<td>Salt-and-pepper</td>
</tr>
<tr>
<td>Geometric transformations &amp; Feature space augmentation</td>
<td>Noise injection</td>
<td>Mixing images datasets</td>
<td>AHCD &amp; Hijja</td>
<td>AHCD &amp; Hijja</td>
<td></td>
</tr>
</tbody>
</table>

Recognized character

Figure 2: Proposed recognition schema for Arabic handwritten characters datasets.
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 backpropagation operations.

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 backpropagation 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:

$$y_i = \frac{\exp(h_i)}{\sum^c_{j=1} \exp(h_j)},$$  

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

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.

$$E = -\sum^p_{j=1} (\hat{y}_j \log(y_j) + (1 + \hat{y}_j)\log(1 - y_j)),$$  

where $\hat{y}_j$ 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

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

where $\varphi$ 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 $\varphi$. $\alpha$ is the model learning rate. A too-large value of $\alpha$ may lead to the divergence of the GD algorithm and may cause the oscillation of the model performance. A too-small $\alpha$ stops the learning process.

5.2.2. Gradient Descent with Momentum [43]. The momentum hyperparameter $m$ defines the velocity by which the learning rate $\alpha$ 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:
\[
\phi(t + 1) = \phi(t) - \alpha(t) \frac{\partial E}{\partial \phi(t)}
\]  
(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

\[
E[G^2(t)] = \gamma E[G^2(t-1)] + (1 - \gamma)G^2(t),
\]
\[
\Delta \theta(t) = \frac{1}{\sqrt{E[G^2(t)] + \epsilon}} G(t),
\]
\[
\phi(t + 1) = \phi(t) - \alpha \Delta \theta(t),
\]
where \(E[G^2(t)]\) is the decaying average over past squared gradients and \(\gamma\) 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:

\[
E[G^2(t)] = 0.9 E[G^2(t-1)] + 0.1 G^2(t),
\]
\[
\phi(t + 1) = \phi(t) - \alpha \Delta \theta(t).
\]

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

\[
r(t) = \beta_1 r(t-1) + (1 - \beta_1) \frac{\partial E}{\partial \phi(t)}
\]
\[
v(t) = \beta_2 v(t-1) + (1 - \beta_2) \left( \frac{\partial E}{\partial \phi(t)} \right)^2
\]
where \(r(t)\) and \(v(t)\) are the 1st and the 2nd order moments vectors, \(\beta_1\) and \(\beta_2\) 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 \(v(t)\):

\[
\tilde{r}(t) = \frac{r(t)}{(1 - \beta_1^t)}
\]
\[
\tilde{v}(t) = \frac{v(t)}{(1 - \beta_2^t)}
\]
where \(\beta_1^t\) is \(\beta_1\) power \(t\) and \(\beta_2^t\) is \(\beta_2\) power \(t\).

The Adam update equation is expressed as follows:

\[
\phi(t + 1) = \phi(t) - \alpha \frac{\tilde{r}(t)}{\sqrt{\tilde{v}(t)} + \epsilon}
\]
(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 \( t \) \( (\frac{\partial E}{\partial \phi(t)}) \) :

\[
v(t) = \beta_2 v(t-1) + (1 - \beta_2) \left| \frac{\partial E}{\partial \phi(t)} \right|^2 .
\]

(19)

The generalization of this update to the \( \ell_p \) norm is as follows:

\[
v(t) = \beta_2^p v(t-1) + (1 - \beta_2^p) \left| \frac{\partial E}{\partial \phi(t)} \right|^p .
\]

(20)

To avoid being numerically unstable, \( \ell_1 \) and \( \ell_2 \) norms are most common in practice. However, in general \( \ell_{\infty} \) also shows stable behavior. As a result, the authors propose AdaMax and demonstrate that \( v(t) \) with \( \ell_{\infty} \) converges to the more stable value. Here,

\[
u(t) = \beta_2^\infty v(t-1) + (1 - \beta_2^\infty) \left| \frac{\partial E}{\partial \phi(t)} \right|^\infty
\]

\[
= \max(\beta_2, v(t-1), \left| \frac{\partial E}{\partial \phi(t)} \right|).
\]

(21)

\[
\varphi(t + 1) = \varphi(t) - \alpha \frac{v(t)}{u(t)}.
\]

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

\[
v(t) = \gamma v(t-1) + \alpha \left( \frac{\partial E}{\partial \phi(t)} - \gamma v(t-1) \right),
\]

(22)

\[
\varphi(t + 1) = \varphi(t) - \alpha v(t).
\]

The update equation using Nadam is expressed as follows:

\[
r = \left( \beta_1 r(t) + \frac{1 - \beta_1}{1 - \beta_1} \frac{\partial E}{\partial \phi(t)} \right),
\]

(23)

\[
\varphi(t + 1) = \varphi(t) - \frac{\alpha}{\sqrt{v(t) + \varepsilon}} r.
\]

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 cost 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 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:
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:

\[
A = \frac{TP + TN}{TP + TN + FN + FP}. \tag{24}
\]

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

\[
R = \frac{TP}{TP + FN}. \tag{25}
\]

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

\[
P = \frac{TP}{TP + FP}. \tag{26}
\]

F1 measure is a combination of Recall and Precision measures:

\[
F1 = 2 \times \frac{P \times R}{P + R}. \tag{27}
\]

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
two convolution layers with 16 filters of size \((3 \times 3)\), then the remaining 2 convolution layers are with 32 filters of size \((3 \times 3)\), and each two convolution layers are followed by max-pooling layers with \((2 \times 2)\) 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 96.73%, recall of 96.67%, and an F1 score of 96.67%, while CNN-for-AHCD achieved an average overall test set accuracy of 96.73%, precision of 96.73%, recall of 96.73%, and an F1 score of 96.72%.

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

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%.

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

Acc. (train) 0.88 0.88 0.88 0.88 0.88 0.88 0.88 0.88 0.88 0.88 0.88 0.88
Acc. (test) 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80
Macro avg 0.81 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80
Weighted avg 0.81 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80

**Figure 9:** AUC curve of Arabic handwritten characters dataset. (a) AUC of AHCD testing dataset. (b) AUC of Hijja testing dataset.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AHCD Accuracy (%)</th>
<th>Hijja Accuracy (%)</th>
<th>AHCD Precision (%)</th>
<th>Hijja Precision (%)</th>
<th>AHCD Recall (%)</th>
<th>Hijja Recall (%)</th>
<th>AHCD F1 score (%)</th>
<th>Hijja F1 score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>99.76</td>
<td>96.66</td>
<td>97.60</td>
<td>88.46</td>
<td>99.76</td>
<td>96.67</td>
<td>97.60</td>
<td>88.46</td>
</tr>
<tr>
<td>SGD</td>
<td>99.44</td>
<td>95.68</td>
<td>94.16</td>
<td>87.85</td>
<td>99.45</td>
<td>95.68</td>
<td>94.16</td>
<td>87.85</td>
</tr>
<tr>
<td>RMSprop</td>
<td>99.72</td>
<td>96.22</td>
<td>96.55</td>
<td>87.98</td>
<td>99.73</td>
<td>96.30</td>
<td>94.28</td>
<td>87.93</td>
</tr>
<tr>
<td>AdaGrad</td>
<td>88.62</td>
<td>83.54</td>
<td>63.90</td>
<td>60.37</td>
<td>88.73</td>
<td>83.78</td>
<td>64.95</td>
<td>83.54</td>
</tr>
<tr>
<td>Nadam</td>
<td>99.77</td>
<td>96.73</td>
<td>97.76</td>
<td>88.57</td>
<td>99.77</td>
<td>96.80</td>
<td>97.57</td>
<td>87.86</td>
</tr>
<tr>
<td>Momentum</td>
<td>99.16</td>
<td>95.56</td>
<td>94.16</td>
<td>87.76</td>
<td>99.18</td>
<td>95.72</td>
<td>94.11</td>
<td>87.73</td>
</tr>
<tr>
<td>AdaMax</td>
<td>99.89</td>
<td>96.19</td>
<td>97.19</td>
<td>87.57</td>
<td>99.90</td>
<td>96.25</td>
<td>97.01</td>
<td>87.86</td>
</tr>
</tbody>
</table>
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 (١) will be the same as the character Noon isolated (٧).

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
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 demonstrates 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 images.
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
- ADDBase: 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
- $@$: Convolution operation
- $C$: Convolution map
- $a$: Size of convolution map
- $S_\cdot$: 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
- $P_i$: Output of the convolution layer
- $P_{i-1}$: Convoluted image
- $Y_i$: Output of the $i$th fully connected hidden layer
- $Y_{i-1}$: Output of the $(i-1)$th fully connected hidden layer
- $H_{i}$: Weight sum vector
- $B$: Bias
- $E$: Cost function
- $\hat{y}$: Desired output
- $\phi$: Update of the filter $F$
- $\frac{\partial E}{\partial \phi}$: Gradient
- $\alpha$: Model learning
- $m$: Momentum
- $v(t)$: Moment gained at the $i$th iteration
- $\varepsilon$: Smoothing value
- $G(t)$: Sum of the squares of the gradient
- $E[G(t)^2]$: Decaying average
- $r(t)$: Moments vector
- $\beta$: 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**


Computational Intelligence and Neuroscience 15

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


