Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05070-8

ORIGINAL ARTICLE

Arabic handwriting recognition system using convolutional neural network

Najwa Altwaijry[1] · Isra Al-Turaiki[2]

Received: 11 October 2019 / Accepted: 3 June 2020

Abstract
Automatic handwriting recognition is an important component for many applications in various fields. It is a challenging problem that has received a lot of attention in the past three decades. Research has focused on the recognition of Latin languages' handwriting. Fewer studies have been done for the Arabic language. In this paper, we present a new dataset of Arabic letters written exclusively by children aged 7–12 which we call Hijja. Our dataset contains 47,434 characters written by 591 participants. In addition, we propose an automatic handwriting recognition model based on convolutional neural networks (CNN). We train our model on Hijja, as well as the Arabic Handwritten Character Dataset (AHCD) dataset. Results show that our model's performance is promising, achieving accuracies of 97% and 88% on the AHCD dataset and the Hijja dataset, respectively, outperforming other models in the literature.

Keywords Convolutional neural network · Arabic character recognition · Hijja Dataset · Machine learning

1 Introduction

Automatic handwriting recognition can be defined as the ability of a system to identify human handwritten input. The handwriting can be from many sources, such as paper documents, images, touch-screens, or other devices. Handwriting that is input through scanning is considered offline, while input through a pen tip is considered online [1]. Handwriting recognition is considered a challenging problem in computer vision. Normally, there are variations in the handwriting of different individuals. In addition, the handwriting of a single writer may be slightly different each time [32].

✉ Najwa Altwaijry
ntwaijry@ksu.edu.sa

Isra Al-Turaiki
ialturaiki@ksu.edu.sa

1 Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

2 Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

The main difficulties in handwriting recognition revolve around distortions and pattern variability, so feature extraction is of paramount importance. Selecting features manually may lead to insufficient information being available to accurately predict the character class. However, a large number of features generally cause problems due to increased dimensionality.

Handwriting recognition problem has been studied using many methods, such as: support vector machines (SVMs), K-nearest neighbors (KNNs), neural networks (NNs), and, recently, convolutional neural networks (CNNs) [2, 10, 33]. However, previous research in this field has focused on the recognition of handwriting in Latin languages [9]. Arabic language is the most widely spoken Semitic language, with an estimated 300 million speakers [30]. It is ranked as the fifth most prevalent spoken language in the world. Handwriting recognition for Arabic is still an open problem for many reasons. Arabic is written with the cursive Arabic alphabet from right to left. The Arabic alphabet contains 28 letters, each of which has 2–4 different forms depending on the letter's position within a word. In addition, Arabic writing makes use of diacritical marks that represent short vowels and other sound, for example: *fat-ha*, *dhumma,* and *kasra*. Arabic also has many *ligatures*, formed by combining two or more letters, such as: *alif-laam.*

Most handwriting recognition models have focused on number recognition as compared to alphabetic characters, as well as adult handwriting. Recognizing children's handwriting is becoming an important part of many applications as children are increasingly exposed to and use technology, such as smartphones and tablet devices, both for entertainment and educational purposes. Handwriting using a finger or stylus is becoming one of the preferred user input options. To the best of our knowledge, no work has been done on children's handwriting.

In this paper, we present the Hijja[1] dataset of handwritten Arabic letters. Hijja is a free publicly available dataset of single letters, collected from Arabic-speaking school children between the ages of 7 and 12 in Riyadh, Saudi Arabia. It represents a total of 47,434 characters written by 591 participants in different forms. Hijja is expected to contribute to the research in Arabic handwriting recognition. In particular, it can be used for the development of educational applications for children. In addition, we describe a model for offline Arabic handwriting recognition using deep neural networks. The architecture of CNNs is inherently suitable for problems with high dimensionality, as in the case of images. Moreover, CNNs have shown success in digit recognition problems as reviewed in the literature [10]. The proposed model is evaluated using two datasets, our Hijja dataset and the AHCD dataset [17], then compared to similar approaches.

The rest of this paper is organized as follows: Sect. 2 provides a brief background on CNNs. Section 3 discusses related works in the literature of handwriting recognition techniques. In Sect. 4, we present Hijja, our dataset of handwritten Arabic letters. Section 5 presents our proposed work. Section 6 shows our model's performance compared to other models in the literature, and finally, Sect. 7 concludes this paper.

2 Background: convolutional neural networks

Traditional optical character recognition (OCR) systems lacked efficiency, as character features were hard-coded and used to match characters. On the other hand, neural networks are able to learn features by analysing a dataset, alleviating the need for manual hard coding of features. At the end of the training process, the neural network learns parameters, making it more flexible when handwriting styles are changed. The creation of the Modified National Institute of Standards and Technology (MNIST) dataset of 70,000 handwritten characters in 1998 ushered a new era of research into handwriting recognition. In fact, LeCun's [26] paper drew attention to the field of neural networks for handwriting recognition, and with the advent of powerful GPUs, CNNs are now capable of achieving excellent results at handwriting recognition [11].

Deep Learning is a field in machine learning that employs representation learning through expressing the input data in multiple levels of simpler representations [19]. In other words, complex concepts such as "car" or "person" are built through layers of simpler concepts such as contour, or edges.

A CNN is a special type of deep learning neural networks that has shown unprecedented success in image-related problems. There are many applications of CNNs, including image classification, image semantic segmentation, object detection in images, etc. [36].

The input data in a CNN are processed in a grid-like topology [19]. For example, an image can be thought of as a grid of size $D \times D \times C$ of input values, where $D \times D$ is the number of pixels in the image and C is number of channels per pixel (1 for a grayscale or 3 for RGB). This input grid is passed to a number of layers: convolutional layers, pooling layers, and fully connected layers. A simple CNN that can recognize handwritten digits (0–9) is shown in Fig. 1. We describe each layer briefly in the following subsections.

2.1 Convolutional layer

The convolution in CNNs comes from the mathematical convolution operation, although it is defined differently than it is defined in mathematics or engineering. Convolution is applied to at least one layer of the NN. A convolutional layer consists of a number of filters K (also called kernels). Each filter serves as a feature identifier, detecting features such as: corners, edges, or endpoints. A filter is a grid of size $N \times N \times R$, where N is the height and width of the filter, and R is the number of image channels C. Each filter slides (or convolves) over all locations in the input grid, multiplying each pixel by the corresponding value in the filter. Then, multiplications are all added to produce a single number. Each convolution produces a feature map, as shown in Fig. 2. The result of applying K convolutional filters is a set of K feature maps, each of size $D - N + 1$.

Feature maps are passed to another mathematical operation called activation function. This is done to allow for the classification of linearly separable classes. A number of activation function exist in the literature, such as the Sigmoid function, the rectified linear unit (ReLU) function, or the tanh function.

[1] Hijja is available at https://github.com/israksu/Hijja2.

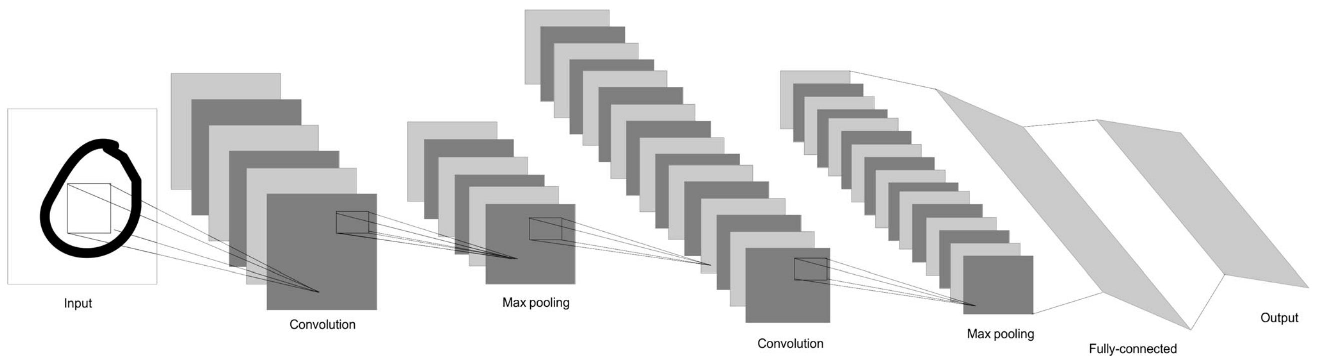
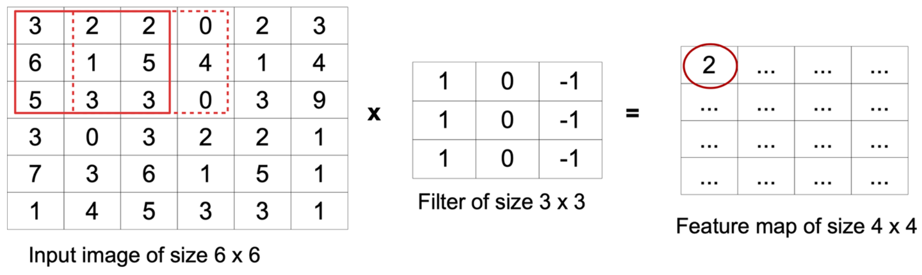


Fig. 1 An example of a simple CNN architecture for classifying handwritten digits

Fig. 2 Illustration of applying a convolution filter of size 3 × 3 to produce a feature map of size 4 × 4



3	2	2	0	2	3
6	1	5	4	1	4
5	3	3	0	3	9
3	0	3	2	2	1
7	3	6	1	5	1
1	4	5	3	3	1

Input image of size 6 x 6

x

1	0	-1
1	0	-1
1	0	-1

Filter of size 3 x 3

=

2	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

Feature map of size 4 x 4

2.2 Pooling layer

The convolutional layer is usually followed by a pooling layer. Pooling is applied to each feature map to reduce the resolution of the image. There are many ways to apply pooling; such as using mean or max with a window of size $q \times q$. Figure 3 shows an example of max pooling.

2.3 Fully connected layer

After the convolutional layers and pooling layers, there may be any number of fully connected layers. A fully connected layer is composed of many neurons, where each neuron is connected to all neurons in the adjacent layers. These layers at the end of network are used to make predictions by the network and to provide the final nonlinear combinations of features.

A major concern in machine learning is reducing the error on unseen examples. A model that performs well on training data, but badly on unseen test data, is not desirable. The methods used to reduce these errors are collectively called regularization methods. In CNNs, dropout and weight decay are frequently used regularization methods.

3 Related work

There are a number of datasets for Arabic characters: the IFN/ENIT database [15], developed by the Institut für Nachrichtentechnik and the Ecole Nationale d'Ingénieurs de Tunis, consists of 26,549 images of Tunisian town/village names written by 411 people. The database of Arabic handwriting database (AHDB) [3] contains images of words describing numbers and quantities in checks. The Center for Pattern Recognition and Machine Intelligence (CENPARMI) has a number of databases for Arabic characters, such as the Database for Recognition of Handwritten Arabic Cheques [4], and the database for Arabic off-line handwriting recognition [6]. In addition, the authors in [18] presented a new database containing 16,800 Arabic characters. The Center for Microprocessor Applications for Training Education and Research Database CMATERDB 3.3.1 Arabic handwritten digit dataset [20] consists of 3000 unique 32 × 32 pixel RGB bitmap images, making it a source of 3000 unique samples. The AHCD dataset [17] is composed of 16,800 characters written by 60 participants; the age range is between 19 and 40 years.

Many efforts have been made to address the problem of Arabic handwriting recognition. For Arabic numeral recognition, Das et al. [14] developed a set of 88 features representing samples of handwritten Arabic numerals. A multi-layer perceptron (MLP) classifier was developed with an input layer, a single hidden layer, and an output layer. The MLP was trained with the back propagation (BP) algorithm and used for classification of Arabic numerals using the CMATERDB 3.3.1 Arabic handwritten digit dataset [20]. Experimental results on the database of 3000 samples showed that the model achieved an average accuracy of 94.93%.

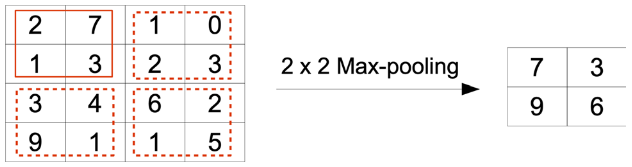


Fig. 3 Illustration of applying max pooling with a window of size 2×2

To improve the recognition performance of Arabic numerals, Ashiquzzaman et al. [8] proposed two methods. First, they introduced two changes to the MLP-based model [14]: the use of rectified linear unit (ReLU) as the activation of each neuron in input layer and hidden layer and the softmax function in the outer classifying layer. Second, they used CNN trained using back-propagation algorithm. All the models were evaluated using the CMATERDB 3.3.1 Arabic handwritten digit dataset [20]. The proposed CNN method outperforms both the updated MLP and the original MLP-based model. The latter two produced identical accuracy, 93.8%, while the CNN obtained 97.4%.

Restricted Boltzmann machine (RBM) has also been utilized for Arabic handwriting recognition. In [7], an RBM was used to extract features followed by a CNN to classify digits on the CMATERDB 3.3.1 Arabic handwritten digit dataset, achieving an accuracy of 98.59%.

In 2018, Latif et al. [24] proposed a CNN architecture for the recognition of handwritten of mixed numerals of multiple languages: Eastern Arabic, Persian, Devanagari, Urdu, and Western Arabic. The input layer is equivalent to the image size which is 28*28 pixels, followed by two hidden convolution layers where a 5*5 kernel window size is used. Both convolution layers are then followed by maxpool layers with a kernel of 2*2. The overall accuracy of the combined Multilanguage database was 99.26% with a precision of 99.29% on average. The average accuracy of each individual language was found to be 99.322%.

Recently, Ashiquzzaman et al. [9] introduced changes to the CNN proposed in [8]. The authors used data augmentation to the CMATERDB 3.3.1 dataset, and changed the activation function from ReLU to ELU. Experimental results showed that the proposed method of CNN obtained an accuracy of 99.4%. This is better than the performance of models presented in [8] and [14], where they achieved accuracy values of 93.8% and 97.4%, respectively.

For the recognition of characters and words, Mars and Antoniadis [28] constructed an Arabic dataset containing 6090 characters and 1080 words. They implemented a neural network approach using time delay neural networks (TDNN) [31] as their classifier, achieving an accuracy of 98.50% for letters and 96.90% for words. Maalej et al. [27] proposed a hybrid CNN-BLSTM model for Arabic handwriting recognition. The CNN is used for automatic feature extract from raw images. Then, the bidirectional long short-term memory (BLSTM) followed by a connection is temporal classification layer (CTC) is used for sequence labelling. The experiments conducted on IFN/ENIT Database showed that the recognition rate of the hybrid model reaches 92.21%.

El-Sawy et al. [18] trained a CNN and tested it on the AHCD database that contains 16,800 handwritten Arabic characters. Their model has two convolutional layers, as well as employing dropout and normalization. The proposed CNN achieved 94.9% accuracy on the testing data.

Alaasam et al. [5] used a CNN to recognize historical Arabic handwritten text. The CNN consisted of: two convolutional layers, two fully connected layers, and one final fully connected layer. Experiments were conducted on three datasets: historical handwritten text images, synthesized text image dataset, and the printed text image dataset. The best accuracy, 85%, was obtained for combination of the handwritten text image dataset and the synthesized text image dataset.

Mudhsh et al. [29] proposed an Alphanumeric very deep neural network (VGG net) for the recognition of Arabic handwritten alphanumeric characters. The model was constructed using 13 convolutional layers, 2 max-pooling layers, and 3 fully-connected layers. Two regularization methods, dropout and augmentation, were used to avoid overfitting. Experiments were conducted on two different databases: the ADBase database (a database of Arabic handwritten digits from 0 to 9) and the HACDB database (a database of handwritten Arabic characters). The model achieved an accuracy of 99.57% for the ADBase database and 97.32% for the HACDB database.

Younis [37] designed a CNN for handwritten Arabic character recognition. The proposed CNN consisted of three convolutional layers followed by a fully connected layer. Experimental results showed that the CNN was able to achieve 94.7% and 94.8% accuracy, using the AHCD and the AIA9K datasets, respectively.

Table 1 summarizes the literature reviewed on automatic handwriting recognition. From the previous literature, we observe that the focus has been on number recognition as compared to alphabetic characters. In addition, models for Arabic handwriting recognition have been trained exclusively on adult handwriting. To the best of our knowledge, no work has been done on children's handwriting. Recognizing children's handwriting is becoming an important part of many applications as children are increasingly exposed to and use technology.

Table 1 Summary of handwriting recognition models using deep learning

References	Year	Model	Dataset	Type	Size	Accuracy
Das et al. [14]	2010	MLP	CMATERDB	Digits	3000	94.93%
Mars et. al. [28]	2016	TDNN	Mars DB	Chars	6090	98.5%
				Words	1080	96.90%
Ashiquzzaman et al. [8]	2017	MLP	CMATERDB	Digits	3000	93.8%
		CNN				97.4%
Alani [7]	2017	RBM-CNN	CMATERDB	Digits	3000	98.59%
El-Sawy et al. [18]	2017	CNN	AHCD	Chars	16,800	94.9%
Alaasam et al. [5]	2017	CNN	HAHPT	Words	96,289	85%
Mudhsh et al. [29]	2017	Very deep NN	ADBase	Digits	70,000	99.57%
			HACDB	Chars	6600	97.32%
Maalej et al. [27]	2018	CNN-BLSTM	IFN/ENIT	Words	32,492	92.21%
Younis [37]	2018	CNN	AHCD	Chars	16,800	94.7%
			AIA9K	Chars	8737	94.8%
Latif et al. [24]	2018	CNN	MADBase	Digits	70,000	99.32%
			MNIST	Digits	70,000	99.32%
			HODA	Digits	80,000	99.32%
			Urdu	Digits	8500	99.32%
			DHCD	Digits	20,000	99.32%
			Combined	Digits	20,000	99.26%
Ashiquzzaman et al. [9]	2019	CNN	CMATERDB	Digits	3000	99.4%

4 Hijja dataset

Datasets are an essential part of any computer vision system. A benchmark dataset allows researchers to compare different machine learning approaches and techniques quickly and fairly [12]. In this section, we describe the Hijja[2] dataset of handwritten Arabic letters. Hijja is a free publicly available dataset of single letters, collected from Arabic-speaking school children between the ages of 7 and 12. Data were collected in Riyadh, Saudi Arabia, from January, 2019 to April, 2019. It represents a total of 47,434 characters written by 591 participants in different forms. Next, we explain the data acquisition process, data cleaning, and preprocessing, and, finally, provide a detailed description of the dataset.

4.1 Data collection

To collect the data from children, a (12*9) data collection matrix was printed on A4 paper, as shown in Fig. 4(a). Arabic contains 28 unique characters, while the matrix represents 108 Arabic cursive character forms based on various letter positions: disconnected character, character at the beginning of a word, character at the end of a word, and character in the middle of a word. Most Arabic letters are connected to the letter that immediately follows, with a few exceptions that are not connected, even if they appear in the middle of the word.

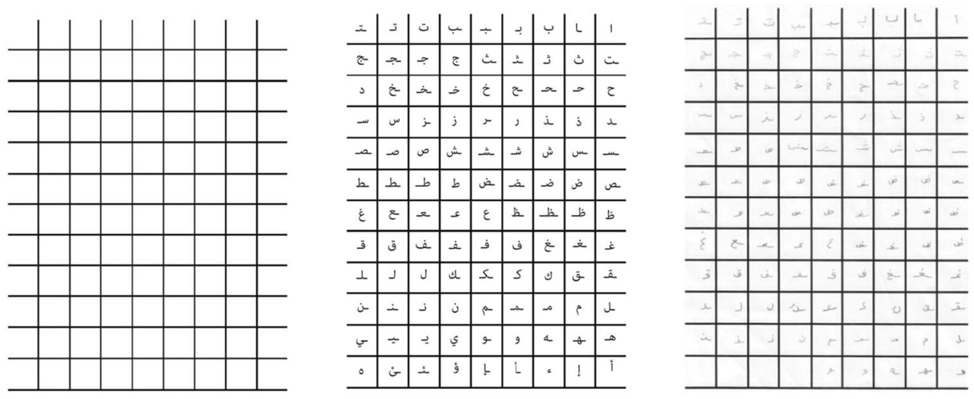
Children were asked to fill the empty matrix in Fig. 4a following the order shown in the reference matrix, as shown in Fig. 4b. After acquiring consent from participating schools, teachers were provided with copies of the data collection matrix and the reference matrix. One thousand data collection matrices were distributed among school children, and 862 matrices were obtained from participants, of which only 591 were suitable for scanning. The 591 obtained matrices were scanned with a resolution of 300dpi in grayscale. An example of the raw scanned data is shown in Fig. 4c. Of these 591 matrices, many were partially full, with the upper portion of the matrix containing the first half of the alphabet filled in and the remainder of the matrix empty.

4.2 Data cleaning and preprocessing

Collecting data from children is a challenging process. The children had trouble following the reference paper, resulting in missing letters, letters entered in wrong positions, and repeated letters. Marks and visible erased pencil strokes were frequent and had to be manually cleaned. Conversely, some pencil strokes were faded and had to be darkened. Tilted scanned papers were manually rotated. The scanned raw PNG image was then segmented into 108 square PNG images, each of which contained a single letter

[2] Hijja is available at https://github.com/israksu/Hijja2.

Fig. 4 Data matrices



(a) Empty Data Collection Matrix **(b)** Data Reference Matrix **(c)** Sample Filled Matrix

form (see Fig. 4b) of size 256*256 pixels, which were further resized into 32*32 pixels using Python.

As previously mentioned, many filled matrices were partially full, resulting in an unbalanced representation of letters, with more PNG files for the first half of the alphabet than for the second, in general. The expected number of images from 591 scanned matrices would be $591 * 108 = 63,828$; however, after discarding the unfilled matrix cells, we randomly selected a subset of PNG files from each letter for a more balanced dataset, for a total of 47,434 characters, see Table 2.

The dataset is organized into 29 folders, representing 29 classes, each corresponding to an Arabic letter, with one folder for the "hamza". Each folder contains subfolders of the various letter forms for each letter. Each subfolder contains the images for that particular letter form. Table 2 shows the statistics of characters per character form in each folder. Vocalization diacritics that mark vowels and other sounds that cannot be represented by Arabic letters (harakat) are not included in our dataset.

5 Deep neural network model

This section outlines the neural network used for Arabic character recognition. Any pattern recognition method usually involves three steps: preprocessing, feature extraction, and classification [21]. Preprocessing the data employs cleaning the data is some way, allowing it to be passed into the next stage, feature extraction, where relevant information that aids in classification is extracted. Feature extraction is a difficult process, and is usually time consuming and cannot process raw visual data. The third step is classification, where either the manually extracted features are used to classify the image into a specific class using an appropriate classifier [13], or some form of automatic feature extraction may be employed, such as SVMs [35] and CNNs [25].

In this section, we discuss our deep neural network and the methods implemented to improve the recognition performance on the Hijja dataset.

5.1 Input layer

The input layer is an $H * W * D$ pixel image, where H is the height, W is the width, and D is the depth, in pixels. A colored RGB image has $D = 3$, while a grayscale image has $D = 1$. Our model has as input a $32 * 32 * 1$ grayscale image.

5.2 Hidden layers

Hidden layers in a CNN are comprised of convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform feature extraction on the input image, extracting features such as corners, edges, or endpoints, then applying an appropriate non-linear function that allows us to avoid collapsing the network into one linear function. Our model is composed of three convolutional layers, where each layer performs a batch normalization operation after an ReLU activation function.

Our architecture was inspired by a number of CNN architectures in the literature. Our input image is 32×32 pixels, and so, a smaller filter size, i.e., 3×3, is more appropriate for this image. A filter can be intuitively seen as a feature detector. The number of filters should roughly correspond with the number of features that the layer is trying to detect. Our input is a small grayscale image, and through trial and error, an initial number of 64 filters were found to work well. As we wish to keep the representational power of the network, we increase the number of feature maps as the networks deepen after each pooling layer. We apply padding in all convolutional layers to

Table 2 Character data description. B: beginning of word, E: end of word, M: middle of word, and D: disconnected letter

Character	Form	Files	Form	Files
ʾalif	B, D, no hamza	456	E, M, no hamza	443
	B, D, with top hamza	460	B, D, with bottom hamza	460
	E, M, with top hamza	464	E, M, with bottom hamza	458
bāʾ	D	452	B	452
	M	449	E	446
tāʾ	D	451	B	447
	M	459	E	439
ṯāʾ	D	454	B	461
	M	471	E	441
ǧīm	D	472	B	458
	M	463	E	464
ḥāʾ	D	462	B	459
	M	456	E	467
ḫāʾ	D	471	B	466
	M	454	E	455
dāl	D, B	449	M, E	438
ḏāl	D, B	429	M, E	425
rāʾ	D, B	424	M, E	423
zāy	D, B	426	M, E	434
sīn	D	440	B	427
	M	425	E	437
šīn	D	439	B	412
	M	436	E	420
ṣād	D	438	B	427
	M	437	E	421
ḍād	D	432	B	402
	M	433	E	426
ṭāʾ	D	452	B	423
	M	527	E	351
ẓāʾ	D	422	B	425
	M	501	E	358
ʿayn	D	434	B	435
	M	442	E	425
ġayn	D	427	B	430
	M	422	E	442
fāʾ	D	434	B	420
	M	458	E	420
qāf	D	435	B	422
	M	438	E	447
kāf	D	438	B	429
	M	438	E	431
lām	D	437	B	442
	M	437	E	434
mīm	D	435	B	432
	M	425	E	434
nūn	D	440	B	449
	M	436	E	453
hāʾ	D	428	B	436
	M	437	E	433
wāw	D, B	439	M, E	435

Table 2 continued

Character	Form	Files	Form	Files
yāʾ	D	435	B	426
	M	434	E	433
hamza	E	425	M, E with wāw	430
	M with yāʾ	426	E with yāʾ	427

overcome the problems of image shrinkage and information loss around the perimeter of the image.

The first convolutional layer has as input a $32 * 32 * 1$ grayscale image. We use $k = 64$ $(3 * 3 * 1)$ kernels, zero-padding of 1, and a stride of 1, for a convolutional layer of size $[((32 + 2 * 1) - 3)/1] + 1 = 32$, i.e., a layer of $32 * 32 * 64$. Each activation map i is calculated as shown in Eq. 1, where l is the current layer, $B_i^{(l)}$ is a bias matrix, $k^{(l-1)}$ is the number of kernels used in the previous layer, W is the current layer kernel matrix, and $Y^{(l-1)}$ is the output of the previous layer. Our nonlinearity is the ReLU function, defined as shown in Eq. 2:

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{k^{(l-1)}} W_i^{(l)} Y_j^{(l-1)} \tag{1}$$

$$ReLU(x) = max(0, x). \tag{2}$$

The first convolutional layer is followed by a max-pooling layer that has a $2 * 2 * 1$ window size, resulting in a $16 * 16 * 64$ layer. This is followed by a local response normalization layer [23].

The next convolutional layer has $k = 128$ feature maps with a kernal size of $3 * 3 * 64$, zero-padding of 1, and a stride of 1, for a convolutional layer of size $[((16 + 2 * 1) - 3)/1] + 1 = 16$, i.e., a layer of $16 * 16 * 128$. The non-linearity is an ReLU activation function. This is followed by a max-pooling layer that has a $2 * 2 * 1$ window size, resulting in a $8 * 8 * 128$. This is followed by a local response normalization layer.

The final convolutional layer has $k = 256$ feature maps with a kernal size of $3 * 3 * 128$, zero-padding of 1, and a stride of 1, for a convolutional layer of size $[((8 + 2 * 1) - 3)/1] + 1 = 8$, i.e., a layer of $8 * 8 * 256$. The non-linearity is an ReLU activation function. This is followed by a max-pooling layer that has a $2 * 2 * 1$ window size, resulting in a $4 * 4 * 256$. This is followed by a local response normalization layer.

The tensor is then flattened into a 4096 neuron fully connected layer, then two other fully connected layers, with sizes 1024 and 512, respectively. All fully connected layers use a 80% dropout rate to reduce overfitting [34], set experimentally.

5.3 Output layer

The output layer is a 29 class Softmax layer (one class for each Arabic letter, plus the hamza) on the Hijja dataset. On the AHCD dataset, the output layer is a 28 class softmax layer. Softmax outputs a probability-like prediction for each character class, see Eq. 3, where N is the number of output classes:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}}. \tag{3}$$

5.4 Optimization

In our model, we tested two optimizers: Stochastic Gradient Descent and Adam [22], and selected Adam as it was found to work better. The loss function used is the categorical cross-entropy loss, which is widely used to calculate the probability that the input belongs to a particular class. It is usually used as the default function for multi-class classification. In our model, we set the learning rate to $lr = 0.001$, set experimentally.

A summary of our model is shown in Fig. 5. We compare our proposed model with CNN-for-AHCD [18]. CNN-for-AHCD consists of two convolutional layers. As the input image size in the AHCD dataset is 32×32 pixels and the authors did not use any padding, the first convolutional layer size is $28 \times 28 \times 80$. This layer is followed by a pooling layer that reduces the size of the convolutional layer to $14 \times 14 \times 80$. The second convolutional layer is of size $10 \times 10 \times 64$ and is followed a pooling layer of size $5 \times 5 \times 64$. This is followed by a fully connected layer of size 1024, which is followed by the classification output layer. The ReLU function is used in all convolutional layers as the activation function, and the pooling layer is a max-pooling layer using a 2×2 pixel window. The Softmax function is applied to the fully connected layer to produce 28 output classes, as per the AHCD dataset.

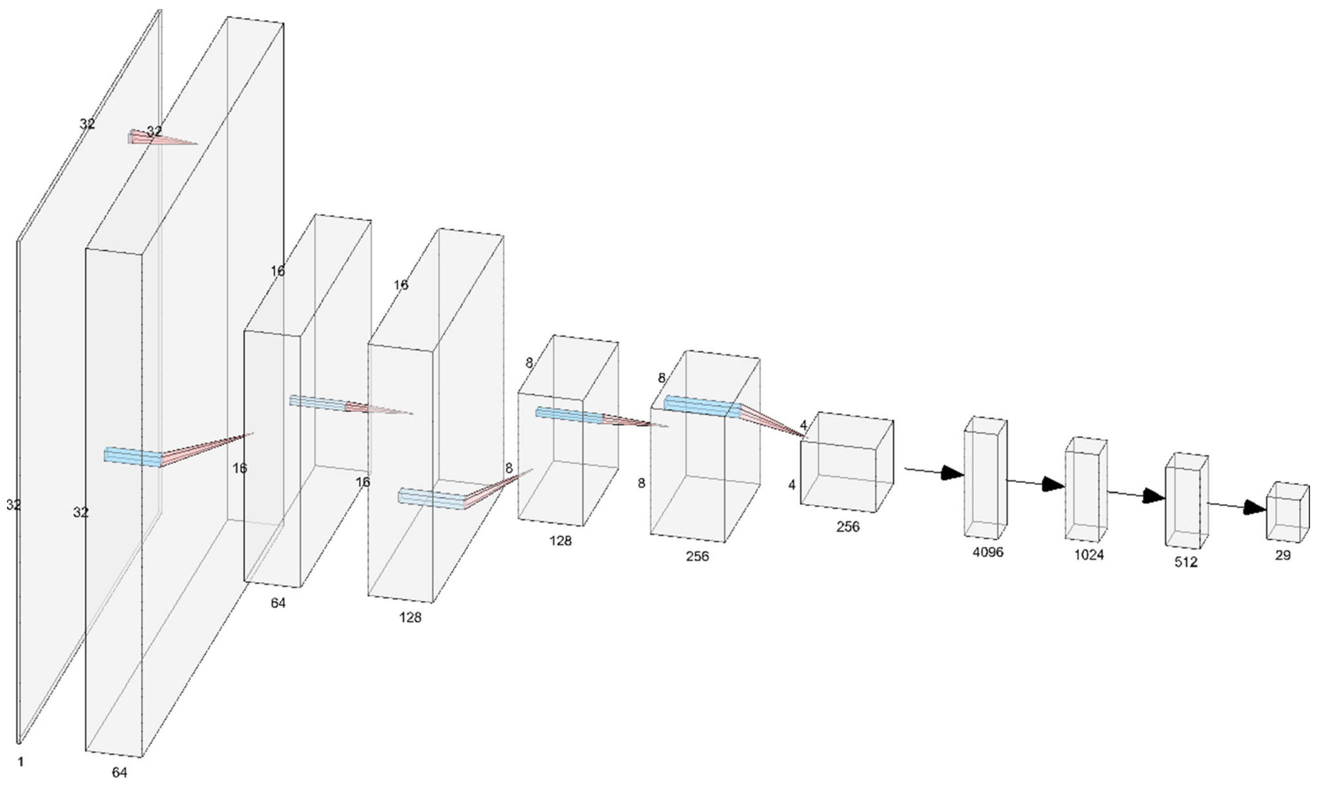


Fig. 5 Deep neural network for Arabic character recognition

6 Experimental results

6.1 Experimental setup

Our model was trained using the Adam optimizer, with a learning rate of 0.001. The model was trained for 30 epochs, set experimentally, on an NVIDIA RTX 2080 Ti GPU. We evaluate the performance of our proposed model using the following measures:

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

$$R = \frac{TP}{TP + FN}. \tag{4}$$

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

$$P = \frac{TP}{TP + FP}. \tag{5}$$

– *F1 measure*: which is a measure that combines Recall and Precision:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}. \tag{6}$$

Here, TP (true positives) is the total number of images that can be correctly labeled as belonging to a class x, FP (false positives) represents the total number of images that have been incorrectly labeled as belonging to a class x, FN (false negatives) represents the total number of images that have been incorrectly labeled as not belonging to a class x; and finally, TN (true negatives), represents the total number of images that have been correctly labeled as not belonging to a class x.

We conduct two experiments to study the performance of our model. In the first experiment, the Hijja dataset is used to train and test both our proposed model and CNN-for-AHCD. We modified the last layer in CNN-for-AHCD to output 29 classes instead of 28 classes, to use it with our dataset. The Hijja dataset was split into 60% for training, 20% for validation, and 20% for testing. We provide the raw data, as well as the CSV files of the train and test sets. In the second experiment, we used the AHCD dataset to train and test both our proposed model and CNN-for-AHCD. The AHCD dataset was also split into 60% for training, 20% for validation, and 20% for testing. To validate our results, both experiments were conducted ten times, and the best classifier, according to the accuracy metric, was used to report the results.

6.2 Results and discussion

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 [16], 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%, precision of 87.88%, recall of 87.81%, and an F1 score of 87.8%, 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%. 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.

On the AHCD dataset, which has 28 classes, our model achieved an average overall test set accuracy of 97%, precision of 96.78%, recall of 96.73%, and an F1 score of 96.73%, 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.

Table 3 Experimental results on the Hijja and AHCD datasets

Character	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. ʾalif	0.93	0.97	0.95	0.99	0.98	0.98	0.96	0.99	0.98	0.99	1.00	1.00
2. bāʾ	0.82	0.91	0.86	0.92	0.97	0.94	0.97	0.97	0.97	0.98	0.99	0.98
3. tāʾ	0.66	0.88	0.75	0.89	0.89	0.89	0.87	0.95	0.91	0.89	0.97	0.93
4. ṯāʾ	0.76	0.81	0.78	0.90	0.87	0.88	0.95	0.88	0.92	0.95	0.96	0.95
5. ğīm	0.79	0.85	0.82	0.89	0.92	0.91	0.95	0.96	0.95	0.97	0.97	0.97
6. ḥāʾ	0.83	0.60	0.70	0.85	0.78	0.81	0.93	0.93	0.93	0.94	0.98	0.96
7. ḫāʾ	0.76	0.77	0.77	0.88	0.84	0.86	0.94	0.93	0.93	0.99	0.97	0.98
8. dāl	0.65	0.69	0.67	0.82	0.74	0.78	0.91	0.94	0.93	0.93	0.96	0.95
9. ḏāl	0.70	0.68	0.69	0.74	0.74	0.74	0.96	0.91	0.93	0.96	0.93	0.94
10. rāʾ	0.86	0.87	0.87	0.83	0.93	0.88	0.89	0.98	0.94	0.95	0.97	0.96
11. zāy	0.87	0.89	0.88	0.87	0.88	0.88	0.94	0.88	0.91	0.94	0.93	0.94
12. sīn	0.84	0.92	0.88	0.93	0.93	0.93	0.95	0.91	0.93	0.99	1.00	1.00
13. šīn	0.86	0.82	0.84	0.90	0.93	0.92	0.92	0.98	0.95	0.99	0.99	0.99
14. ṣād	0.75	0.81	0.78	0.84	0.88	0.86	0.85	0.96	0.90	0.97	0.97	0.97
15. ḍād	0.80	0.76	0.78	0.84	0.86	0.85	1.00	0.89	0.94	1.00	0.96	0.98
16. ṭāʾ	0.90	0.83	0.87	0.92	0.92	0.92	0.96	0.94	0.95	0.95	0.97	0.96
17. ẓāʾ	0.83	0.87	0.85	0.93	0.92	0.93	0.97	0.94	0.95	0.97	0.97	0.97
18. ʿayn	0.74	0.70	0.72	0.79	0.79	0.79	0.95	0.90	0.92	0.98	0.95	0.97
19. ġayn	0.83	0.72	0.77	0.90	0.79	0.84	0.89	0.97	0.93	0.98	0.98	0.98
20. fāʾ	0.77	0.65	0.71	0.74	0.83	0.78	0.93	0.84	0.88	0.94	0.99	0.9
21. qāf	0.81	0.80	0.81	0.88	0.86	0.87	0.87	0.91	0.89	0.99	0.93	0.96
22. kāf	0.86	0.78	0.82	0.87	0.90	0.89	0.98	0.96	0.97	0.98	0.98	0.98
23. lām	0.90	0.87	0.88	0.91	0.94	0.93	0.98	0.97	0.98	1.00	0.98	0.99
24. mīm	0.83	0.85	0.84	0.89	0.90	0.89	0.98	0.98	0.98	0.98	0.98	0.98
25. nūn	0.70	0.77	0.73	0.84	0.82	0.83	0.92	0.92	0.92	0.97	0.91	0.94
26. hāʾ	0.81	0.76	0.78	0.86	0.87	0.86	0.97	0.96	0.96	0.97	0.96	0.97
27. wāw	0.93	0.82	0.87	0.90	0.89	0.89	0.96	0.94	0.95	0.93	0.95	0.94
28. yāʾ	0.82	0.81	0.82	0.95	0.92	0.93	0.97	0.97	0.97	0.99	0.97	0.98
29. hamza	0.74	0.73	0.74	0.84	0.81	0.83	na	na	na	na	na	na
acc. (train)			0.88			**0.93**			0.91			**0.98**
acc. (test)			0.80			**0.88**			0.94			**0.97**
macro avg	0.81	0.80	0.80	0.87	0.87	0.87	0.94	0.94	0.94	0.97	0.97	0.97
weighted avg	0.81	0.80	0.80	0.88	0.88	0.88	0.94	0.94	0.94	0.97	0.97	0.97

We again note that our model outperforms CNN-for-AHCD by a large margin on all metrics. This pattern can be seen on all metrics for each class in Table 3, as well.

Comparing the results of both algorithms on the different datasets, we see that both algorithms had worse performance on the Hijja dataset, as compared to the AHCD dataset. This implies that the Hijja dataset can be considered a harder, more complex, and varied dataset, as it was more difficult for both programs to learn a model that fits the data.

It is particularly interesting to note how the results of the algorithms differ per class on both datasets. The letters (5. ǧīm), (6. ḥāʾ) and (7. ḫāʾ) are more difficult to classify in Hijja than they are in AHCD. This reflects the difficulties that children have in writing these letters, as well as the large variety of acceptable writing forms. As an example, Fig. 6 shows the various forms the letter (6. ḥāʾ) can take.

Many people write the letters (8. dāl) and (9. ḏāl) very similarly to (6. ḥāʾ) and (7. ḫāʾ), respectively, with the reader using the surrounding word (context) to read the letters correctly. This is reflected in the corresponding metrics of these letters. In Fig. 7, 7a shows the letter (8. dāl) in the beginning or disconnected end-of-word, and Fig. 7b shows the letter (6. ḥāʾ) in the beginning of the word, while 7c shows the letter (9. ḏāl) in the beginning or disconnected end-of-word, and 7d shows the letter (7. ḫāʾ) in the beginning of the word.

In addition, (18. ʿayn) and (19. ġayn) are also written similarly to the letters (20. fāʾ) and (21. qāf), respectively, which is reflected in their metrics. Figure 8 shows these letters when found in the middle of the word: Fig. 8a shows the letter (18. ʿayn), Fig. 8b shows the letter (19. ġayn), Fig. 8c shows the letter (20. fāʾ), and Fig. 8d shows the letter (21. qāf).

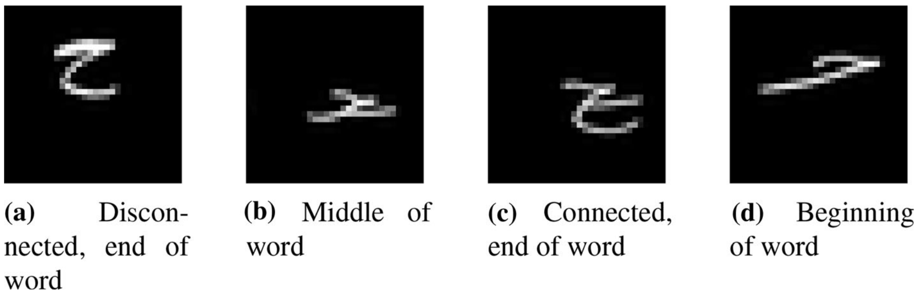


Fig. 6 The letter (6. ḥāʾ) written in various positions

(a) Disconnected, end of word **(b)** Middle of word **(c)** Connected, end of word **(d)** Beginning of word

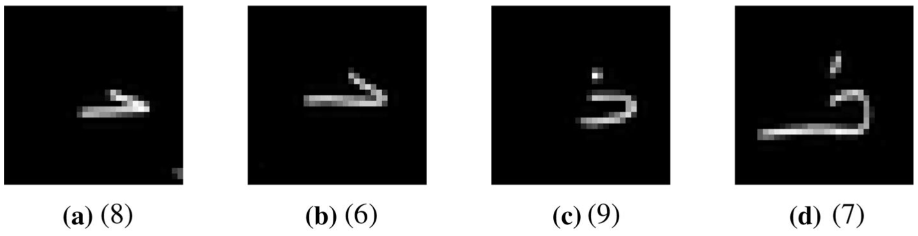


Fig. 7 Letters (8), (9), (6), and (7) written similarly; the number represents alphabetical order

(a) (8) **(b)** (6) **(c)** (9) **(d)** (7)

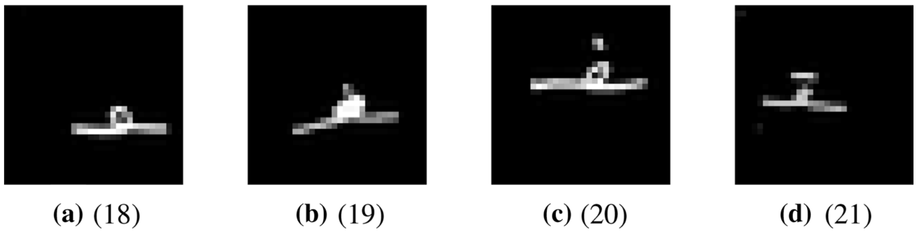


Fig. 8 Letters (18), (19), (20), and (21) written similarly; the number represents alphabetical order

(a) (18) **(b)** (19) **(c)** (20) **(d)** (21)

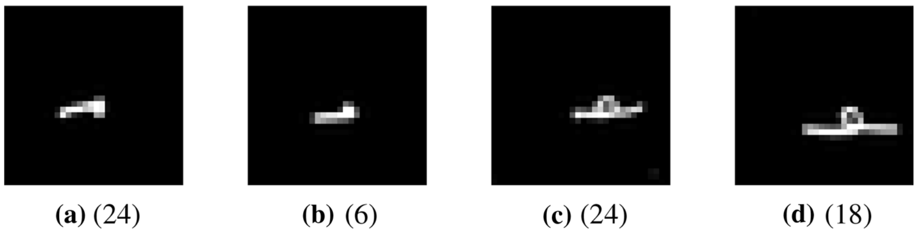


Fig. 9 Letters (24), (6), and (18) written similarly; the number represents alphabetical order

(a) (24) **(b)** (6) **(c)** (24) **(d)** (18)

The letter (24. mīm) is also written similarly to (18. ʿayn) and (6. ḥāʾ) when found in the beginning or middle of the word. (24. mīm) has an F1 score of 0.89 in Hijja, compared to an F1 score of 0.98 in AHCD. Figure 9a and b shows the letters (24. mīm) and (6. ḥāʾ) in the beginning of the word, respectively, while Fig. 9c and d shows the letters (24. mīm) and (18. ʿayn) in the middle of the word, respectively.

7 Conclusion

Automatic handwriting recognition is a challenging problem. In this paper, we presented Hijja, a new dataset for Arabic handwriting. We proposed a deep learning model to recognize handwritten letters. Our model achieved excellent results on the AHCD dataset, and promising results compared to other models on the Hijja dataset. We were interested in creating a dataset that can be used in real-life applications to aid in teaching children handwriting as well as spelling. Many children nowadays use technology, and their particular quirks when it comes to handwriting are different from those of adults. Thus, a system trained on adult handwriting is unlikely to work as well if children were to use the system. To address this lack in available datasets, the Hijja dataset was collected and made available to researchers.

In the future, it would be beneficial to study the various successful machine learning models on this dataset, such as MLP, SVM, and auto-encoders. Our model can be used to recognize children's Arabic handwriting in any application that requires this function. In addition, it is capable of recognizing adult Arabic handwriting. It may be used as part of a pipeline that segments connected Arabic letters, to recognize full words. The model may be used in a transfer learning environment to recognize characters in other languages. We plan to incorporate our neural network into an application for children that teaches Arabic spelling.

Acknowledgements This research project was supported by a grant from the Research Center of the Female Scientific and Medical Colleges, the Deanship of Scientific Research, King Saud University. This research has been approved by the "Standing Committee on Ethics of Scientific Research" at King Saud University (No. KSU-KSU-HE-19-363).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abdalkafor AS (2018) Survey for databases on Arabic off-line handwritten characters recognition system. In: 2018 1st International conference on computer applications information security (ICCAIS), pp 1–6
2. Ahmad I, Fink GA (2016) Class-based contextual modeling for handwritten Arabic text recognition. In: 2016 15th International conference on frontiers in handwriting recognition (ICFHR), pp 554–559
3. Al-Ma'adeed S, Elliman D, Higgins CA (2002) A data base for Arabic handwritten text recognition research. In: Proceedings eighth international workshop on frontiers in handwriting recognition, IEEE. pp 485–489
4. Al-Ohali Y, Cheriet M, Suen C (2003) Databases for recognition of handwritten Arabic cheques. Pattern Recognit 36(1):111–121
5. Alaasam R, Kurar B, Kassis M, El-Sana J (2017) Experiment study on utilizing convolutional neural networks to recognize historical Arabic handwritten text. In: 2017 1st International Workshop on Arabic script analysis and recognition (ASAR), pp 124–128
6. Alamri H, Sadri J, Suen CY, Nobile N (2008) A novel comprehensive database for Arabic off-line handwriting recognition. In: Proceedings of 11th International conference on frontiers in handwriting recognition, ICFHR, vol. 8, pp 664–669
7. Alani A (2017) Arabic handwritten digit recognition based on restricted boltzmann machine and convolutional neural networks. Information 8(4):142
8. Ashiquzzaman A, Tushar AK (2017) Handwritten Arabic numeral recognition using deep learning neural networks. In: 2017 IEEE International conference on imaging, vision and pattern recognition (icIVPR), IEEE. pp 1–4
9. Ashiquzzaman A, Tushar AK, Rahman A, Mohsin F (2019) An efficient recognition method for handwritten Arabic numerals using CNN with data augmentation and dropout. In: Balas VE, Sharma N, Chakrabarti A (eds) Data management, analytics and innovation, advances in intelligent systems and computing. Springer, Singapore, pp 299–309
10. Baldominos A, Sáez Y, Isasi P (2019) A survey of handwritten character recognition with mnist and emnist. Appl Sci 2019:3169
11. Ciresan DC, Meier U, Gambardella LM, Schmidhuber J (2011) Convolutional neural network committees for handwritten character classification. In: 2011 International conference on document analysis and recognition, pp 1135–1139. IEEE
12. Cohen G, Afshar S, Tapson J, van Schaik A (2017) Emnist: Extending mnist to handwritten letters. In: 2017 International Joint conference on neural networks (IJCNN), pp 2921–2926. IEEE
13. Cruz RM, Cavalcanti GD, Ren TI (2010) Handwritten digit recognition using multiple feature extraction techniques and classifier ensemble. In: 17th International conference on systems, signals and image processing, pp 215–218

14. Das N, Mollah AF, Saha S, Haque SS (2010) Handwritten Arabic numeral recognition using a multi layer perceptron. CoRR **abs/1003.1891**. http://arxiv.org/abs/1003.1891
15. Dimauro G, Impedovo S, Modugno R, Pirlo G (2002) A new database for research on bank-check processing. In: Proceedings eighth international workshop on frontiers in handwriting recognition, pp 524–528. IEEE
16. El-Sawy A, Loey M, Hazem E Convolutional neural network for Arabic handwritten characters dataset classification. https://www.kaggle.com/mloey1/cnn-for-arabic-handwritten-characters/comments. Accessed: 30 May 2019
17. El-Sawy A, Loey M, Hazem E (2017) Arabic handwritten characters dataset. https://www.kaggle.com/mloey1/ahcd1. Accessed: 30 May 2019
18. El-Sawy A, Loey M, Hazem E (2017) Arabic handwritten characters recognition using convolutional neural network. WSEAS Trans Comput Res 5:11–19
19. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press. http://www.deeplearningbook.org
20. Google code archive - long-term storage for google code project hosting. https://code.google.com/ archive/p/cmaterdb/downloads. Accessed: 30 May 2019
21. Impedovo S, Mangini FM, Barbuzzi D (2014) A novel prototype generation technique for handwriting digit recognition. Pattern Recognit 47(3):1002–1010
22. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
23. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
24. Latif G, Alghazo J, Alzubaidi L, Naseer MM, Alghazo Y (2018) Deep convolutional neural network for recognition of unified multi-language handwritten numerals. In: 2018 IEEE 2nd International workshop on Arabic and derived script analysis and recognition (ASAR), pp 90–95
25. Lauer F, Suen CY, Bloch G (2007) A trainable feature extractor for handwritten digit recognition. Pattern Recognit 40(6):1816–1824
26. LeCun Y, Bottou L, Bengio Y, Haffner P et al (1998) Gradient-based learning applied to document recognition. Proc IEEE 86 (11):2278–2324
27. Maalej R, Kherallah M (2018) Convolutional neural network and blstm for offline Arabic handwriting recognition. In: 2018 International Arab conference on information technology (ACIT), Werdanye, Lebanon, 2018, pp 1–6. https://doi.org/10.1109/ACIT.2018.8672667
28. Mars A, Antoniadis G (2016) Arabic online handwriting recognition using neural network. Int J Artif Intell Appl(IJAIA) 7(5)
29. Mudhsh MA, Almodfer R (2017) Arabic handwritten alphanumeric character recognition using very deep neural network. Information 8(3)
30. Owens J (2013) The Oxford handbook of Arabic linguistics. Oxford University Press, Oxford
31. Peddinti V, Povey D, Khudanpur S (2015) A time delay neural network architecture for efficient modeling of long temporal contexts. In: Sixteenth annual conference of the international speech communication association
32. Porwal U, Shi Z, Setlur S (2013) Chapter 18—Machine learning in handwritten Arabic text recognition. In: Rao CR, Govindaraju V (eds) Handbook of statistics, handbook of statistics, vol 31. Elsevier, Amsterdam, pp 443–469
33. Ramzan M, Khan HU, Awan SM, Akhtar W, Ilyas M, Mahmood A, Zamir A (2018) A survey on using neural network based algorithms for hand written digit recognition. Int J Adv Comput Sci Appl. https://doi.org/10.14569/IJACSA.2018.090965
34. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2005) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
35. Szarvas M, Yoshizawa A, Yamamoto M, Ogata J (2005) Pedestrian detection with convolutional neural networks. In: IEEE Proceedings. intelligentvehicles symposium, 2005, Las Vegas, NV, USA, pp 224–229. https://doi.org/10.1109/IVS.2005.1505106
36. Wu J (2017) Introduction to convolutional neural networks. National Key Lab for Novel Software Technology vol 5, Nanjing University, China, p 23
37. Younis K (2018) Arabic handwritten characters recognition based on deep convolutional neural networks. Jordan J Comput Inform Technol (JJCIT) 3