Neural Computing and Applications (2024) 36:12103–12115
https://doi.org/10.1007/s00521-024-09674-2

ORIGINAL ARTICLE

DeepAHR: a deep neural network approach for recognizing Arabic handwritten recognition

Helala AlShehri[1]

Received: 21 September 2023 / Accepted: 25 March 2024 / Published online: 19 April 2024

Abstract
Automatic handwritten character recognition plays a significant role in various applications across multiple fields. With the growing interest in automatic handwriting recognition and the advancement of deep learning methods, researchers have achieved significant improvements in the development of English handwriting recognition methods. However, the recognition of Arabic handwriting has received insufficient attention. In this paper, a novel "DeepAHR" model is presented to accurately and efficiently recognize Arabic handwritten characters using deep learning techniques. The "DeepAHR" model is based on a convolutional neural network (CNN) and is trained using two recent public datasets: Hijaa and Arabic handwritten characters dataset (AHCD). The overall accuracies of the proposed model were 98.66% and 88.24% on the AHCD and Hijaa datasets, respectively.The experimental results showed that DeepAHR outperformed state-of-the-art methods in the literature. These promising results provide evidence of the successful use of the DeepAHR model for recognizing handwritten Arabic characters

Keywords Arabic handwritten letter · Recognition · CNN · Machine learning · Deep learning

1 Introduction

Automatic handwritten character recognition (AHCR) is important for various applications. There are a multitude of sources for handwriting, such as images, paper documents, and touch screens [1], and there is a growing demand for an accurate application of handwriting recognition that can be used with different source types. The AHCR is characterized as the system's capabilities to recognize handwritten input images [2]. It uses character recognition technology to convert characters into their corresponding digital characters, thereby providing a method for automatically recognizing text in images. AHCR is considered a challenging task because the handwriting of most people differs. Moreover, individual writers' handwriting abilities can change significantly over time [3].

Over the last few decades, AHCR has been an active area of research, and many AHCR methods have been developed to identify different languages. The most common languages are Chinese [4, 5], English [6, 7], and French [8]. Arabic is one of the languages most commonly spoken worldwide, with more than 315 million native speakers. Arabic character recognition has recently received research attention [9]. Recognizing Arabic characters poses a significant challenge in the fields of computer vision and pattern recognition because of the unique characteristics of the Arabic language, such as its distinct spelling, grammar, and pronunciation, compared to other languages [10]. Arabic comprises 28 characters and is typically written in a semi-cursive style from right to left, where the letters are interconnected in a continuous flow. Arabic characters can exhibit four distinct forms based on their position within a word: beginning, middle, end, or standalone. Furthermore, the similarity in shape among Arabic letters presents a difficulty [11]. Table 1 presents the variations in Arabic letters depending on their position in words. It can be noticed that, for instance, the letters "ba," "ta," and "tha " share a noticeable similarity despite being presented in four different positions within a word. Given the variability in

✉ Helala AlShehri
shehrihel@rcjy.edu.sa

[1] Computer and Information Technology Department, Jubail Industerial College, Rd Number 6, Al Huwaylat, Jubail Industrial City, Kingdom of Saudi Arabia

Table 1 Twenty-eight different Arabic alphabet shapes

Alphabet	Isolated form	Beginning	Middle	End
alif	أ	أ	ـأ	ـأ
ba	ب	بـ	ـبـ	ـب
ta	ت	تـ	ـتـ	ـت
thaa	ث	ثـ	ـثـ	ـث
gim	ج	جـ	ـجـ	ـج
haa	ح	حـ	ـحـ	ـح
kha	خ	خـ	ـخـ	ـخ
dal	د	د	ـد	ـد
thal	ذ	ذ	ـذ	ـذ
ra	ر	ر	ـر	ـر
zay	ز	ز	ـز	ـز
sin	س	سـ	ـسـ	ـس
shin	ش	شـ	ـشـ	ـش
sad	ص	صـ	ـصـ	ـص
dad	ض	ضـ	ـضـ	ـض
da	ط	طـ	ـطـ	ـط
za	ظ	ظـ	ـظـ	ـظ
ayn	ع	عـ	ـعـ	ـع
gayn	غ	غـ	ـغـ	ـغ
fa	ف	فـ	ـفـ	ـف
qaf	ق	قـ	ـقـ	ـق
kaf	ك	كـ	ـكـ	ـك
lam	ل	لـ	ـلـ	ـل
mim	م	مـ	ـمـ	ـم
non	ن	نـ	ـنـ	ـن
ha	ه	هـ	ـهـ	ـه
Waw	و	و	ـو	ـو
Ya	ي	يـ	ـيـ	ـي

character shapes depending on the context within a word, automated handwriting recognition of Arabic characters is considerably more complex than that of other languages.

Advancements in deep learning have enabled convolutional neural networks (CNNs) to demonstrate a outstanding ability to identify handwritten characters in various languages, such as Latin, Chinese, Devanagari, and Malayalam [12, 13]. Researchers have enhanced CNN architectures to improve the recognition performance of handwritten characters [13, 14]. This enhancement typically involves fine-tuning CNN hyperparameters, selecting appropriate optimization algorithms [15] and along with utilizing a substantial training dataset [16, 17]. In this study, a new deep CNN model called DeepAHR was developed to recognize handwritten Arabic characters. The proposed DeepAHR was thoroughly tested using two public benchmark datasets: Arabic handwritten characters dataset (AHCD) [2] and Hijaa [18]. The results and comparisons show that this method outperforms the state-of-the-art methods. The main contributions of this study are as follows:

- Reviewing state-of-the-art research in Arabic handwritten character recognition.
- Developing an effective Arabic handwritten character recognition model based on a CNN.
- Investigating and analyzing the impact of different regularization techniques and hyperparameters on the performance of the proposed CNN method.
- A comprehensive method evaluation using two benchmark datasets is provided and compared with state-of-the-art methods.

The remainder of this paper is organized as follows: Sect. 2 presents the related work, and Sect. 3 details the proposed method. Section 4 presents the experimental details, results, and discussion. The conclusions and future work are presented in Sect. 5.

2 Related work

Recently, researchers have developed various techniques to improve Arabic handwritten character recognition results based on CNNs. El-Sawy et al. [18] found that CNN methods outperformed other approaches for feature extraction and classification, particularly with large datasets. However, available handwritten Arabic datasets include only a limited number of images. Therefore, the authors released the AHCD, which was collected from 60 participants aged 19–40 years. The authors proposed a model based on a CNN that achieved 94.9% accuracy on the AHCD. Similarly, Altwaijry and Al-Turaiki [2] released Hijaa dataset containing samples produced by children aged 7–12 years. The researchers introduced a CNN-based system for Arabic handwriting recognition and compared its performance with that of El-Sawy et al. [18]. The empirical results revealed that Altwaijry and Al-Turaiki's model achieved 97% and 88% accuracies for the AHCD and Hijaa datasets, respectively. These results demonstrate that the proposed CNN outperformed the El-Sawy et al. model [18]. Balaha et al. [19] created a complex and extensive Arabic handwriting dataset known as HMBD. They proposed two CNN approaches, HMB1 and HMB2, employing different optimization, regularization, and dropout methods. HMB1 and HMB2 were evaluated on three datasets (AIA9k, CMATER, and HMBD) in 16 experiments. The best results were 98.4%, 97.3%, and 90.7% for AIA9k, CMATER, and HMBD, respectively. Furthermore, the study revealed that data augmentation helped reduce overfitting and increased accuracy.

Ahmed et al. [20] designed a CNN that employed dropout regularization and batch normalization layers to extract optimal features. To assess the effectiveness of the model, the authors evaluated the model on a set of six benchmark datasets: MADBase (digits), SUST-ALT (digits), CMATERDB (digits), SUST-ALT (characters), HACDB (characters), and SUSTALT (names). The model achieved 99% accuracy; however, the model was not evaluated on AHCD. Younis [21] built a CNN with three convolutional layers and a fully connected layer with an overfitting regularization parameter. Experimental results revealed that the proposed approach achieved accuracies of 94.8% and 94.7% on the AIA9K and AHCD datasets, respectively. AlJarrah et al. [22] constructed a CNN model and examined the impact of data-augmentation techniques on its performance. Their results revealed that the model accuracy increased from 97.2% to 97.7% after applying data augmentation to the AHCD dataset. Elleuch et al. [23] proposed a deep belief neural network (DBNN) for recognizing handwritten Arabic characters and words. Their results demonstrated a model accuracy of 97.9% using the HACDB dataset. Elagamy et al. [24] designed a customized CNN handwritten Arabic character recognition approach utilizing deep learning. The proposed approach was evaluated on AHCD, achieving an accuracy rate of 98.54%. Momeni and BabaAli [25] introduced two distinct transformer architectures, namely the transducer and standard sequence to sequence, and assessed their effectiveness in terms of speed and accuracy on the KFUPM handwritten Arabic text (KHATT) dataset [26]. Similarly, in [27], a light encoder–decoder transformer approach was presented for handwritten text recognition, and in [28], an end-to-end method utilizing pre-trained image and text transformers methods for word-level text recognition was suggested.

Several studies have investigated the use of transfer learning to propose solutions for Arabic character handwriting recognition. Alyahya et al. [29] studied the effect of applying the ResNet-18 architecture, and the model was trained and evaluated on AHCD. The best accuracy achieved was 98.3%, utilizing a standard ResNet-18 model. Similarly, the model achieved accuracies of 98.03% and 98.00% by combining ResNet-18 with one fully connected layer and using two fully connected layers with ResNet-18, respectively. Mudhsh et al. [30] proposed a VGG-16-based CNN, and it was trained and evaluated using two benchmark datasets: HACDB for character recognition and MADBase for digit recognition. The model achieved accuracies of 97.32% on the HACDB dataset and 99.66% on the MADBase dataset. Al-Tani et al. [31] adopted the ResNet architecture for handwritten Arabic character recognition. Using AHCD, AIA9K, and MADBase, the accuracies achieved using this model were 99.55%, 99.05%, and 99.8%, respectively. Korichi et al. [32] performed various experiments with different CNN architectures, such as VGG-16 and ResNet, combined with regularization techniques, such as data augmentation and dropout. According to their findings, handcrafted features were less effective than CNN-based methods.

3 Materials and methods

This section discusses the techniques and methods utilized to build the DeepAHR system for detecting handwritten Arabic characters.

3.1 Dataset

Two recent and publicly available datasets were used in this study: AHCD [18] and Hijaa [2]. AHCD contains 16,800 handwritten letters gathered from 60 participants aged from 19 to 40 years, with 90% of them being right-handed. In AHCD, the total number of Arabic class labels is 28 (i.e., from the letter "alef" to "yaa"). Each participant provided a set of twenty-eight letters written ten times. A sample of letters in AHCD is shown in Fig. 1. The dataset was partitioned into two sets, with 80% of the characters used as a training set and the remaining 20% used as the test set. In contrast with the test set, which included 3,360 letters split into 120 images each class, the training set had 13,440 characters partitioned into 480 images.

The second dataset was the Hijaa dataset, which is the largest existing dataset for Arabic character recognition. The dataset was collected from Arabic-speaking children aged 7 to 12 years and is consisted of 47,434 characters created by 591 participants. The dataset is partitioned into 29 files corresponding with the 28 Arabic letters (i.e., from letters "alef" to "yaa") and one file for the Hamza. The letters were written in both isolated and connected forms depending on their positions: at the beginning, middle, and end of a word. A sample of letters from the Hijaa dataset is shown in Fig. 2.

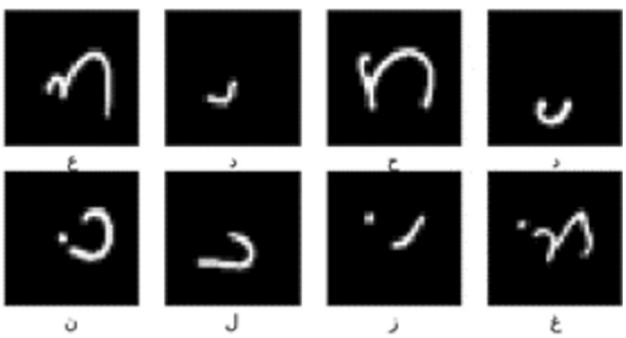

Fig. 1 Samples of Arabic characters in the training set for AHCD

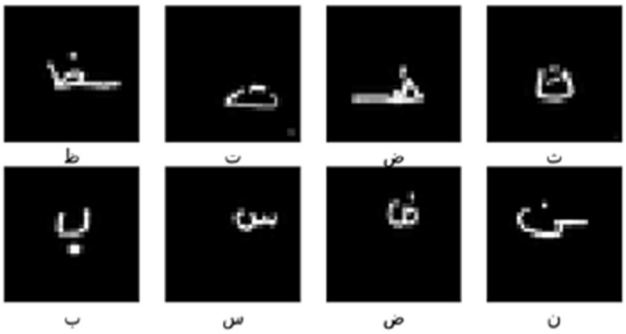

Fig. 2 Samples of Arabic characters in the training set of the Hijaa dataset

3.2 Dataset preprocessing

Data preprocessing is a vital step in preparing data for the best-fitting machine learning model [33]. In this study, the images in AHCD were rotated, as all images were flipped. Figure 3a shows a subset of the images (from AHCD) without any modification, and Fig. 3b shows the same images after transposing. The images in both datasets were normalized by dividing them by 255 and converted into NumPy arrays to use less memory space and increase the training speed. Subsequently, different data augmentation techniques, such as zooming and rotation, were applied to increase the size of the dataset, solve the overfitting problem, and make the model more robust [34]. The details of the data augmentation parameters are listed in Table 2.

3.3 Proposed DeepAHR model

CNNs have proved to be a powerful model for automatic feature extraction and have become state of the art in various image classification problems owing to their high performance in recognizing image patterns. CNNs are a type of deep learning model specifically tailored for analyzing data with a grid-like formation, such as images. They draw inspiration from the structure of the visual cortex in animals [35] and are designed to autonomously and dynamically learn hierarchical spatial features, progressing

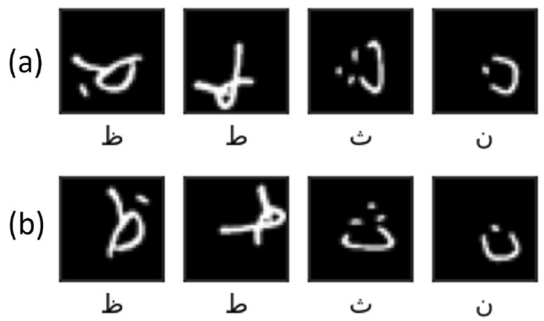

Fig. 3 Sample AHCD letters: **a** before transposing, **b** after transposing

Table 2 Data augmentation techniques with parameter values

Data augmentation technique	Parameter value
Rotation	10
Zooming	0.1
Width shift	0.1
Height shift	0.1

from basic to more complex patterns. CNNs are essentially mathematical frameworks comprising three key types of layers: convolution, pooling, and fully connected layers, as well as an output layer. The convolution and pooling layers focus on extracting features, whereas the fully connected layer is responsible for translating these extracted features into a final output, similar to classification [36, 37]. In CNNs, an image is convolved with filters in the convolution layer to produce feature maps, which are then forwarded to the succeeding layers to extract a complex feature from the input image.

This study proposes a new CNN model called DeepAHR, which is composed of five convolution layers and two fully connected layers. Furthermore, there are activation, pooling, and batch normalization layers between the convolutional and fully connected layers, as shown in Fig. 4. In this section, the proposed DeepAHR method is discussed.

3.3.1 Input layer

The input layer of a CNN is an $H \times W \times D$ image, where H represents the height, W represents the width, and D represents the depth of the pixels. Our model's input image was a $32 \times 32 \times 1$ gray-scale image representing Arabic characters fed into the input layer. In CNNs, the input layer gives only the shape of the image, without feature extraction. The input layer then feeds the images into the hidden layers.

3.3.2 Hidden layers

In a CNN, the hidden layers are composed of convolutional, pooling, and fully connected layers. Using the input image, the convolutional layers perform feature extraction, where significant information that assists in classification, such as edges, corners, or endpoints, is identified. Our model comprises five convolution layers, each of which uses a leaky rectified linear unit (LeakyReLU) as the activation function. LeakyReLU is based on the popular nonlinear ReLU activation function [38]; however, it adopts a small slope for negative values as an alternative to the use of a flat slope in ReLU [39].

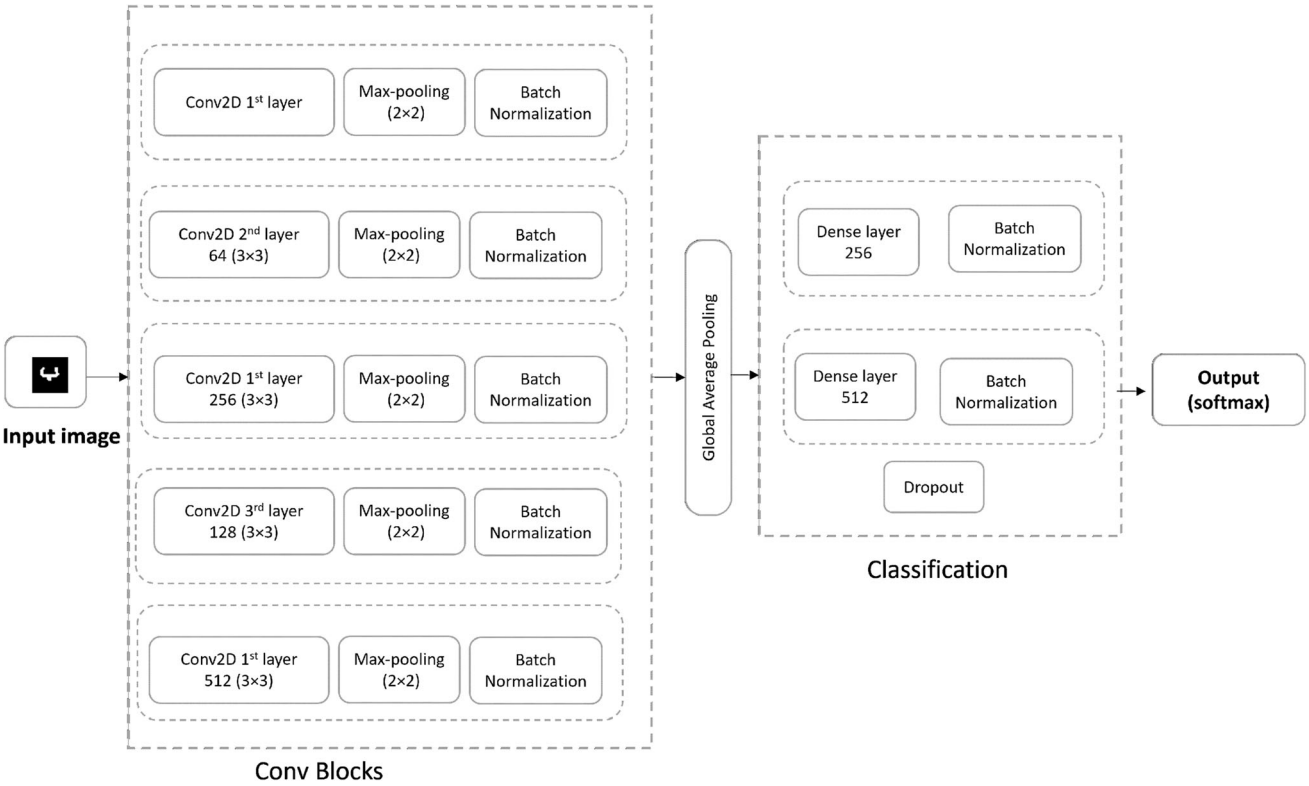

Fig. 4 Proposed DeepAHR model

Each convolutional layer used a small kernel size of 3×3, because the input image size was $32 \times 32 \times 1$ and a smaller filter size was optimal for this classification task. All convolutional layers employ various kernels to generate a feature map for extracting low- and high-level features, such as edges, endpoints, and vertices, from the input image. Furthermore, we used zero padding in each convolutional layer to prevent the loss of information around the image perimeters and problems associated with image shrinking. We also set the stride of the convolution along the height and width of the image to one.

The first convolutional layer used 32 filters, a stride ($s = 1$), the same padding of size 1, and a kernel size of (3×3), yielding an output shape of size $32 \times 32 \times 32$ where the output shape could be computed as $(filters + 2 \times padding - (kernelsize - 1))$. The activation size of the layer was the dot product of the output shapes, giving the first layer an activation size of 32,768 elements. Table 3 lists the activation sizes and output structures of all the layers.

The next four layers are 2D convolutional layers, followed by a max-pooling layer and a batch normalization layer. To keep the network representative, we increased the number of feature maps as the network deepened after each pooling layer. The four subsequent 2D convolutional layers used 64, 128, 256, and 512 filters, respectively. Max-pooling provides the maximum value from the patch of the image covered by the kernel. After each convolutional operation, max-pooling with a $2 \times 2 \times 1$ window size was employed to reduce the size of the features, which aided in lowering network dimensionality. Eliminating insignificant parameters also helped prevent overfitting and decreased computational complexity. Batch normalization is a method for training very deep networks that rescales and recenters the inputs into layers to standardize them. This enhances and accelerates the stability of the learning process during network training while also decreasing the number of epochs required to train the networks. As a result, all convolutional layers other than the first had integrated max-pooling and batch normalization layers. The output of the fifth convolutional layer was fed to the global average pooling layer, which averaged each feature map, and then fed into the fully connected layer, that is, the dense layer.

The final step in the hidden layers of the proposed CNN consisted of two fully connected layers with sizes of 256 and 512 neurons, with all neurons connected to the activation units of the subsequent layer. The fully connected layer was followed by a 40% dropout rate to reduce overfitting, which was selected experimentally.

Table 3 DeepAHR output structure, size, and trainable parameters of the layers

Layer (type)	Output shape	Trainable parameters
Convolution 1	(None, 32, 32, 32)	832
Max pooling 1	(None, 16, 16, 32)	0
Batch normalization	(None, 16, 16, 32)	128
Convolution 2	(None, 16, 16, 64)	18,496
Max pooling 2	(None, 8, 8, 64)	0
Batch normalization	(None, 8, 8, 64)	256
Convolution 3	(None, 8, 8, 128)	73,856
Max pooling 3	(None, 4, 4, 128)	0
Batch normalization	(None, 4, 4, 128)	512
Convolution 4	(None, 4, 4, 256)	295,168
Max pooling 4	(None, 2, 2, 256)	0
Batch normalization	(None, 2, 2, 256)	1,024
Convolution 5	(None, 2, 2, 512)	1,180,160
Max pooling 5	(None, 1, 1, 512)	0
Batch normalization	(None, 1, 1, 512)	2,048
Global average pooling	(None, 512)	0
Fully connected layer 1	(None, 512)	262,656
Fully connected layer 2	(None, 256)	131,328
Dropout	(None, 256)	0
Fully connected layer	(None, 28)	7,196
Total parameters:	1,973,660	
Trainable parameters:	1,971,676	
Non-trainable parameters:	1,984	

3.3.3 Output layer

The output layer employs softmax as an activation function, which classifies the features into multiclasses as required. In AHCD, the output layer is composed of 28 neurons, whereas it is composed of 29 neurons in the Hijaa dataset.

4 Experimental results

4.1 Experimental setup

The implementation and evaluation of the proposed DeepAHR model were conducted using Keras deep learning environments with a TensorFlow backend and a GPU accelerator on Google Colab Pro.

4.1.1 Performance measures

The performance of our proposed model was evaluated using the following measures:

- Accuracy: The ratio of correctly classified images to the total number of predicted images [40]. Equation (1) shows the formula used to compute accuracy:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (1)$$

- Recall: The proportion of correctly classified images among all images in class x [33], computed using Eq. (2):

$$Recall = \frac{TP}{TP + FN}. \quad (2)$$

- Precision: The proportion of correctly classified images among all classified images [40], computed using Eq. (3):

$$Precision = \frac{TP}{TP + FP}. \quad (3)$$

- F1-score: The weighted average of recall and precision [33], computed using Eq. (4):

$$F1 - score = \frac{2Precision \times Recall}{Precision + Recall}. \quad (4)$$

where false positive (FP) represents the total number of images that were incorrectly classified as belonging to class x, true positive (TP) represents the total number of images that could be correctly identified as belonging to class x, false negative (FN) represents the total number of images that were incorrectly classified as not belonging to class x, and true negative (TN) represents the total number of images that could be correctly identified as not belonging to class x.

4.1.2 Training and parameters optimizations

Several attempts were made to tune the network configuration to select the best model that fits both the AHCD and Hijaa datasets. The optimized parameters employed to enhance the performance of the CNN are listed in Table 4. Categorical cross-entropy, which is widely used to measure losses in multiclass label predictions, was employed as the loss function. The model was tested using various numbers

Table 4 Values of parameters employed in the proposed framework during training

Parameter	Value
Loss function	categorical_crossentropy
Batch size	32
Epochs	100
Metrics	Accuracy

of epochs, and the final optimal number of epochs was set to 100. A small batch size of 32 was used, which demonstrated a suitable generalization of the model.

One of the key hyperparameters is the optimizer algorithm, which fits both the AHCD and Hijaa datasets. To determine the best algorithm for the optimizer that fits both datasets, five optimizers were tested: Adam, AdamW, Adagrad, Nadam, and RMSprop with three different learning rates (lr): 0.001, 0.0001, and 0.00001. This totaled 15 experimental models for each dataset and 30 experiments overall. The detailed results of different optimization algorithms are listed in Table 5. The results show that the best accuracy is achieved when adopting the "Nadam" optimizer with a learning rate of 0.001 for both datasets. On AHCD, the proposed model achieved an average overall test set accuracy of 98.66%, recall of 98.66%, precision of 98.68%, and F1-score of 98.66%. For the Hijaa dataset, our model achieved an average overall test accuracy of 88.24%, recall of 91.4%, precision of 91.4%, and F1-score of 91.5%.

The proposed model was trained for over 100 iterations. However, by 35th epoch, the model achieved over 99.05% training accuracy and 98.21% validation accuracy for AHCD. On the Hijaa dataset, after training for 100 epochs, the model achieved a 94.6% training accuracy by 62nd epoch. Therefore, the overall validation accuracy was 91.3% during the validation phase.

Figures 5 and 6 show the training and validation accuracies with respect to epochs on the AHCD and Hijaa datasets, respectively. Figures 5a and 6a show that no overfitting was observed during the training process. From the curve of the loss function (Fig. 5b), it can be observed that the value of the loss starts to drop sharply on AHCD, whereas there are some fluctuations on the Hijaa dataset, as shown in Fig. 6

Table 5 Experimental results using different optimizers and learning rates

Dataset	Learning rate	Optimizers	Accuracy	Precision	Recall	F-measure
AHCD	0.001	Adam	0.9833	0.9835	0.9833	0.9833
		AdamW	0.9845	0.9847	0.9845	0.9845
		Adagrad	0.9458	0.9471	0.945833	0.945833
		RMS_prop	0.9839	0.9842	0.9839	0.9839
		Nadam	**0.9866**	**0.9868**	**0.9866**	**0.9866**
	0.0001	Adam	0.9842	0.9845	0.9842	0.9842
		AdamW	0.9815	0.9818	0.9815	0.9815
		Adagrad	0.9401	0.9419	0.9401	0.9401
		RMS_prop	0.9760	0.9759	0.9760	0.9760
		Nadam	0.9845	0.9848	0.9845	0.9845
	0.00001	Adam	0.9827	0.9830	0.9827	0.9827
		AdamW	0.9818	0.9820	0.9818	0.9818
		Adagrad	0.9440	0.9451	0.9440	0.9440
		RMS_prop	0.9339	0.9363	0.9339	0.9339
		Nadam	0.98482	0.9850	0.9848	0.9848
Hijaa	0.001	Adam	0.8758	0.906	0.906	0.906
		AdamW	0.876	0.908	0.908	0.908
		Adagrad	0.757	0.782	0.782	0.782
		RMS_prop	0.8762	0.908	0.908	0.908
		Nadam	**0.88243**	**0.914**	**0.914**	**0.915**
	0.0001	Adam	0.8781	0.908	0.908	0.908
		AdamW	0.8811	0.912	0.912	0.912
		Adagrad	0.7532	0.779	0.779	0.779
		RMS_prop	0.85643	0.886	0.886	0.886
		Nadam	0.8805	0.912	0.912	0.912
	0.00001	Adam	0.8834	0.915	0.915	0.915
		AdamW	0.88274	0.914	0.914	0.914
		Adagrad	0.75455	0.780	0.780	0.780
		RMS_prop	0.7201	0.743	0.743	0.743
		Nadam	0.8790	0.910	0.910	0.910

Bold text is used to emphasize and highlight specific values, drawing the reader's attention to their exceptional significance

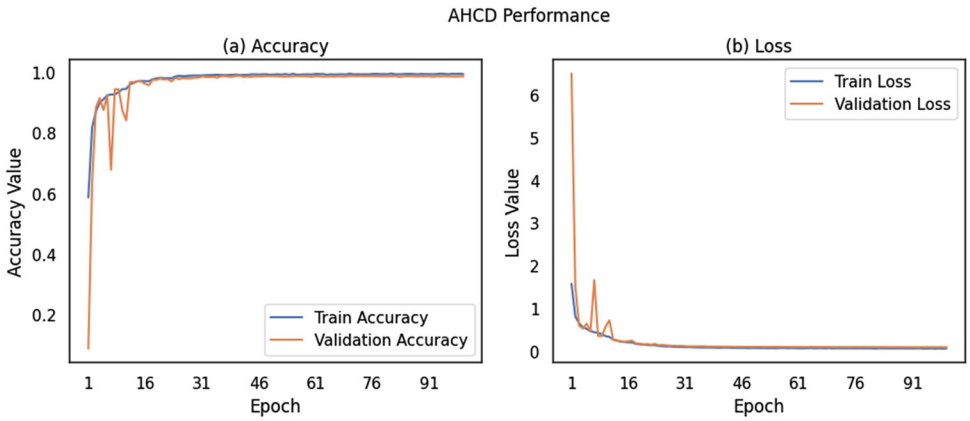

Fig. 5 Training progress for AHCD: **a** training and validation accuracy (higher is better), and **b** training and validation loss (lower is better)

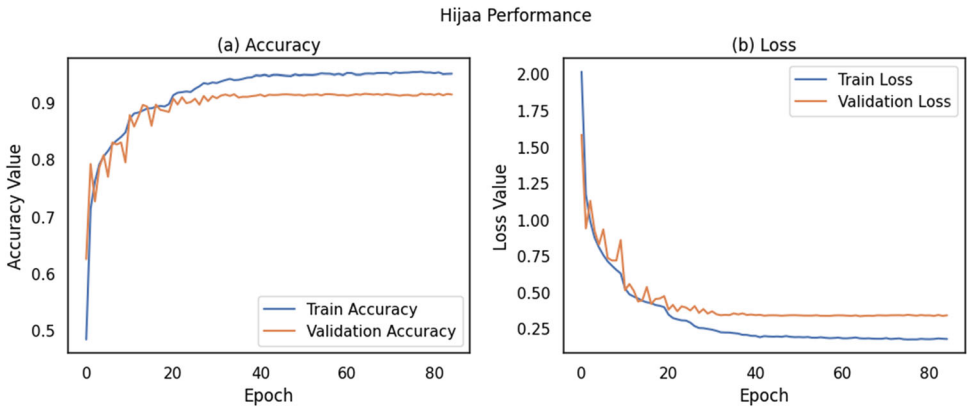

Fig. 6 Training progress for the Hijaa dataset: **a** training and validation accuracy (higher is better), and **b** training and validation loss (lower is better)

4.2 Results and discussion

The classification report, which included the overall performance measures and individual character values (Table 6), shows promising results for the DeepAHR model. It should be noted that all class numbers starting from 0 to 27 correspond to the alphabet from "alif" (ا) to "ya" (ي) for both datasets, whereas number 28 corresponds to the "hamza" (ء) alphabet in the Hijaa dataset.

The overall accuracy of the proposed model for AHCD is 98.66%. The model was evaluated in terms of precision, recall, and F1-score. The average precision, recall, and F1-score were 98.68%, 98.66%, and 98.66%, respectively. For the Hijaa dataset, the overall model accuracy was 88.24%, and the average precision, recall, and F1-score were 91.4%, 91.4%, and 91.5%, respectively.

The results obtained from DeepAHR differed by class for both datasets. The characters 7 (dal "د") and 8 (thal "ذ") are more difficult to recognize in the Hijaa dataset than they are in AHCD. Figure 7 shows various forms that characters 7 (dal "د") and 8 (thal "ذ") can take, as many people write them very similar to letters 5 (haa "ح") and 6 (kha "خ"). Figure 7a shows how character 7 (dal "د"), when positioned at the disconnected end of a word, can be written similarly to character 8 (haa "ح") when positioned at the beginning of an Arabic word, as shown in Fig. 7b. Figure 7c and d shows how characters 8 (Thal "ذ") and 6 (Kha "خ") can be written similarly.

Furthermore, characters 18 (gayn "غ") and 19 (f "ف") are written similarly in the middle of Arabic as shown in Fig. 8. Additionally, Fig. 9 shows how characters 24 (mim "م") and 17 (ayn "ع") can be written similarly when positioned in the middle of a word.

Character 24 (non "ن") is also written similarly to character 8 (thal "ذ"), 6 (kha "خ"), and 10 (Zay "ز") when its position is at the beginning or end of a word, as shown in Fig. 10. This is reflected in its metric, as non "ن" has an F1 score of 0.97 in AHCD, compared with an F1- score of 0.82 in the Hijaa dataset.

4.3 Comparison with existing works

We evaluated our proposed methodology by comparing it with state-of-the-art approaches that focus on recognizing handwritten Arabic characters using the AHCD and Hijaa datasets, as listed in Table 7. Experimental results from AHCD showed that the DeepAHR model outperformed the models used by El-Sawy et al. [18], Younis et al. [21],

Table 6 Classification reports for the AHCD and Hijaa datasets

Arabic Character	Class	AHCD				Hijaa			
		Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
Alif (أ)	0	1	1	1	120	1.00	0.99	0.99	549
Ba (ب)	1	1	0.99	1	120	0.95	0.97	0.96	360
Ta (ت)	2	0.96	0.99	0.98	120	0.92	0.93	0.92	360
Tha (ث)	3	0.99	0.98	0.99	120	0.91	0.94	0.92	366
Gim (ج)	4	0.98	1	0.99	120	0.94	0.95	0.95	372
Haa (ح)	5	0.99	0.98	0.99	120	0.91	0.85	0.88	369
Kha (خ)	6	0.99	0.98	0.99	120	0.92	0.88	0.90	370
Dal (د)	7	0.95	0.99	0.97	120	0.86	0.75	0.80	178
Thal (ذ)	8	0.97	0.94	0.96	120	0.82	0.68	0.75	171
Ra (ر)	9	0.97	0.97	0.97	120	0.92	0.94	0.93	170
Zay (ز)	10	0.97	0.95	0.96	120	0.86	0.94	0.90	172
Sin (س)	11	0.99	0.98	0.99	120	0.96	0.95	0.95	346
Shin (ش)	12	0.99	1	1	120	0.95	0.98	0.96	342
Sad (ص)	13	0.96	1	0.98	120	0.90	0.91	0.91	345
Dad (ض)	14	1	0.97	0.98	120	0.91	0.91	0.91	339
Da (ط)	15	0.97	1	0.98	120	0.95	0.92	0.94	351
Za (ظ)	16	1	0.97	0.98	120	0.92	0.96	0.94	342
Ayn (ع)	17	0.99	0.97	0.98	120	0.85	0.84	0.85	348
Gayn (غ)	18	0.97	0.99	0.98	120	0.87	0.88	0.88	345
Fa (ف)	19	0.95	1	0.98	120	0.85	0.84	0.84	347
Qaf (ق)	20	1	0.95	0.97	120	0.92	0.93	0.92	349
Kaf (ك)	21	0.98	0.99	0.99	120	0.90	0.95	0.92	348
Lam (ل)	22	1	1	1	120	0.92	0.94	0.93	350
Mim (م)	23	0.99	1	1	120	0.93	0.95	0.94	346
Non (ن)	24	0.99	0.96	0.97	120	0.79	0.87	0.82	356
Ha (هـ)	25	0.99	0.97	0.98	120	0.96	0.92	0.94	347
Waw (و)	26	0.98	0.98	0.98	120	0.96	0.94	0.95	175
Ya (ي)	27	0.99	0.99	0.99	120	0.96	0.95	0.95	346
Hamza (ء)	28	–	–	–	–	0.90	0.87	0.88	342
Accuracy		–	–	0.98	3,360	–	–	0.91	9,501
Macro avg		0.98	0.98	0.98	3,360	0.91	0.91	0.91	9,501
Macro avg		0.98	0.98	0.98	3,360	0.91	0.91	0.91	9,501

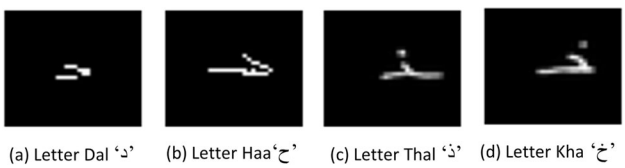

Fig. 7 Different forms of letters (ذ) ,(ح) ,(د), and (خ) written similarly

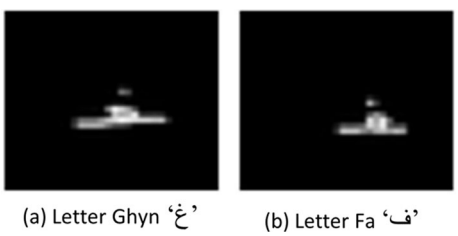

Fig. 8 Letters (غ) and (ف) written similarly

Alyahya et al. [29], and Alheraki et al. [41]. For the experiments conducted on the Hijaa dataset, DeepAHR achieved better results in terms of accuracy than the models used by El-Sawy et al. [18], Younis et al. [21], and Alyahya et al. [29], but not the model used by Alheraki et al. [41]. However, DeepAHR outperformed the model used by Alheraki et al. [41] in terms of recall, precision, and F1-score metrics. Notably, there was a significant difference in

Fig. 9 Letters (م) and (ع) written similarly

the experimental results between the methods on the AHCD and Hijaa datasets. The methods performed poorly on the Hijaa dataset, which implies that the Hijaa dataset presented challenges because it included different forms of each character, including both connected and isolated forms. This represents a higher level of similarity between characters. In addition, the Hijaa dataset was collected from children. In contrast, Arabic characters in AHCD were isolated and collected from adults.

Furthermore, DeepAHR was tested using unseen letters from a test set that produced remarkable results. In this step, eight images were randomly selected from each of the AHCD and Hijaa test datasets. DeepAHR printed the actual and predicted labels for the selected alphabetical images, as shown in Figs. 11, 12.

5 Conclusion

In this study, we proposed a novel "DeepAHR" model for Arabic handwritten character recognition. The "DeepAHR" model is based on a CNN that consists of five convolution layers and two fully connected layers. LeakyReLU was adopted as the activation function for all the layers of the models. Batch normalization was used to enable independent learning for each layer in the model. To determine the best optimizer, five optimizers were tested with three learning rates for each optimizer across 30 experiments on two public datasets: AHCD and Hijaa. The results show that the 'Nadam' optimizer with a learning rate of 0.001 yields the best accuracy for both datasets. We applied data augmentation to address the problem of insufficient handwritten Arabic datasets and improve model generalization. DeepAHR achieved accuracies of 98.66% and 88.24% for AHCD and Hijaa, respectively.

An interesting future direction would be to evaluate the outcomes of alternative augmentation techniques such as generative adversarial networks and adversarial training when applied to an Arabic handwritten letter recognition dataset. In addition, it would be beneficial to create new datasets of different Arabic handwriting styles, such as Naskh, Reqaa, and Kufi. The DeepAHR model could be integrated into various applications, such as digital document processing and automated translation services, to

Fig. 10 Different ways of writing the letter non

Table 7 Comparison between our proposed model and state-of-the-art methods

Dataset	Method	Accuracy	Precision	Recall	F1-score
AHCD	El-Sawy et al. [18]	94.9%	–	–	–
	Younis et al. [21]	97.6%	–	–	–
	Alyahya et al. [29]	98.3%	98.3%	–	–
	Alheraki [41]	97%	97%	97%	97%
	Proposed Model	**98.66%**	**98.68%**	**98.66%**	**98.66%**
Hijaa	Altwaijry et al. [2]	88%	88%	88%	88%
	Alheraki et al. [41]	**91%**	91%	91%	91%
	Proposed Model	88.24%	**91.4 %**	**91.4%**	**91.5%**

Bold text is used to emphasize and highlight specific values, drawing the reader's attention to their exceptional significance

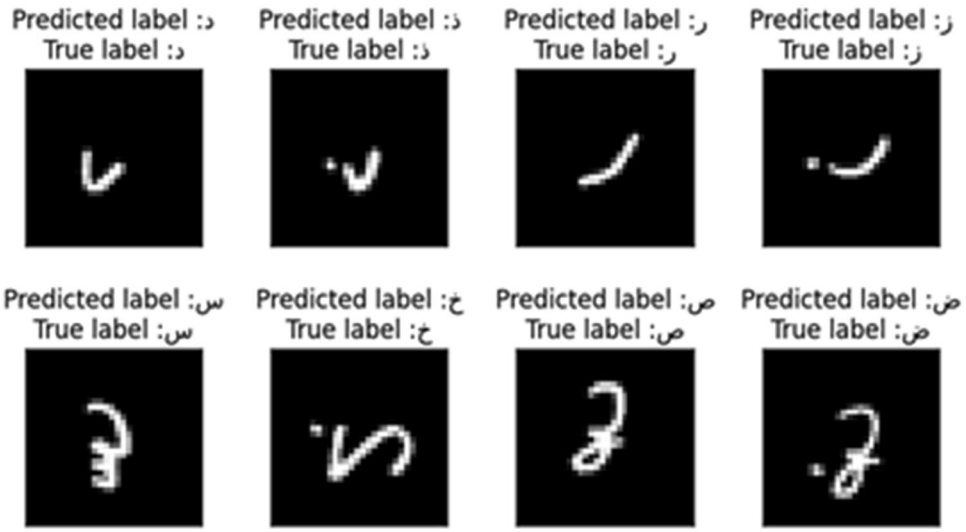

Fig. 11 "DeepAHR" evaluation of Arabic letter images from AHCD

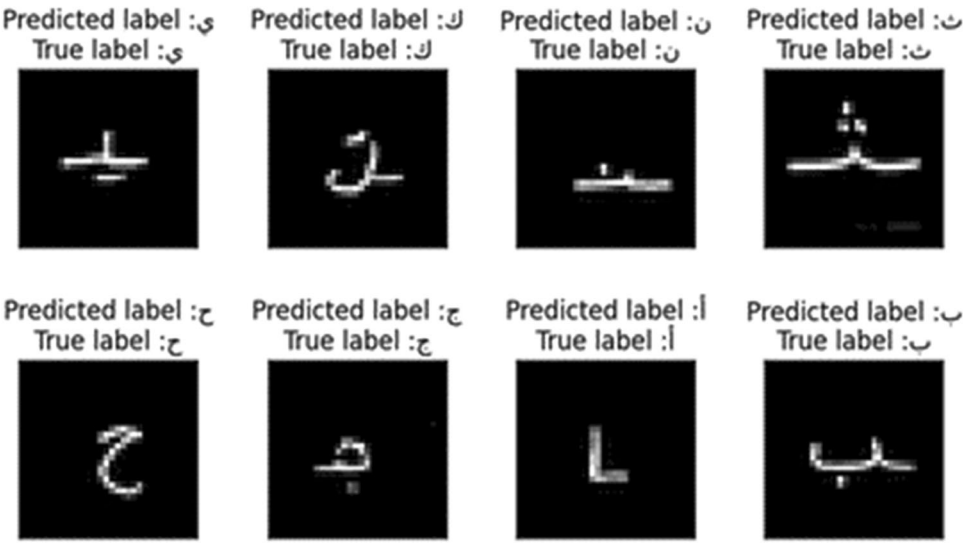

Fig. 12 "DeepAHR" evaluation of Arabic letter images from the Hijaa dataset

enhance their efficiency in handling Arabic handwritten texts.

Funding No funding was received to assist with the preparation of this manuscript

Data availability The data used in this paper are publicly accessible, and details are available in the references section.

Declarations

Conflict of interest The authors have no conflict of interest to declare that is relevant to the content of this article.

References

1. Ali AAA, Mallaiah S (2021) Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout. J King Saud Univ Comput Inf Sci 34:3294–3300
2. Altwaijry N, Al-Turaik I (2021) Arabic handwriting recognition system using convolutional neural network. Neural Comput Appl 33:2249–2261. https://doi.org/10.1007/s00521-020-05070-8
3. Balaha HM, Ali HA, Youssef EK, Elsayed AE, Samak RA, Abdelhaleem MS, Tolba MM, Shehata MR, Mahmoud MR, Abdelhameed MM, Mohammed MM (2021) Recognizing Arabic handwritten characters using deep learning and genetic algorithms. Multimed Tools Appl 80:32473–32509. https://doi.org/10.1007/s11042-021-11185-4
4. Melnyk P, You Z, Li K (2020) A high-performance CNN method for offline handwritten Chinese character recognition and visualization. Soft Comput 24:7977–7987. https://doi.org/10.1007/s00500-019-04083-3
5. Xiao X, Jin L, Yang Y, Yang W, Sun J, Chang T (2017) Building fast and compact convolutional neural networks for offline

handwritten Chinese character recognition. Pattern Recogn 72:72–81. https://doi.org/10.1016/j.patcog.2017.06.032
6. Yuan A, Bai G, Jiao L, Liu Y (2012) Offline handwritten English character recognition based on convolutional neural network. IEEE, pp. 125–129
7. Ptucha R, Such FP, Pillai S, Brockler F, Singh V, Hutkowski P (2019) Intelligent character recognition using fully convolutional neural networks. Pattern Recogn 88:604–613
8. Xiao S, Peng L, Yan R, Wang S (2020) Deep network with pixel-level rectification and robust training for handwriting recognition. SN Comput Sci 1:145. https://doi.org/10.1007/s42979-020-00133-y
9. Shariq M (2015) Arabic and English consonants: a phonetic and phonological investigation. Adv Lang Lit Stud. https://doi.org/10.7575/aiac.alls.v.6n.6p.146
10. Wissam AlKendi LH, Franck Gechter Guyeux C (2024) Advancements and challenges in handwritten text recognition: a comprehensive survey . https://doi.org/10.3390/jimaging10010018 . https://www.mdpi.com/2313-433X/10/1/18
11. Faizullah S, Ayub MS, Hussain S, Khan MA (2023) A survey of OCR in Arabic language: applications, techniques, and challenges. Appl Sci. https://doi.org/10.3390/app13074584
12. Sahu DK, Jawahar CV (2015) Unsupervised feature learning for optical character recognition. In: 2015 13th international conference on document analysis and recognition (ICDAR), pp. 1041–1045. https://doi.org/10.1109/ICDAR.2015.7333920
13. Wagaa N, Kallel H (2020) Vector-based back propagation algorithm of supervised convolution neural network. In: 2020 international conference on control, automation and diagnosis (ICCAD), pp. 1–6 . https://doi.org/10.1109/ICCAD49821.2020.9260520
14. AlJarrah MN, Zyout MM, Duwairi R (2021) Arabic handwritten characters recognition using convolutional neural network. In: 2021 12th international conference on information and communication systems (ICICS), pp. 182–188 . https://doi.org/10.1109/ICICS52457.2021.9464596
15. Soydaner D (2020) A comparison of optimization algorithms for deep learning. Int J Pattern Recognit Artif Intell 34(13):2052013. https://doi.org/10.1142/S0218001420520138
16. Hernández-García A, König P (2019) Further advantages of data augmentation on convolutional neural networks. CoRR, arxiv.org/abs/1906.11052
17. Hidayat AA, Purwandari K, Cenggoro TW, Pardamean B (2020) A convolutional neural network-based ancient sundanese character classifier with data augmentation
18. El-Sawy A, Loey M, El-Bakry H (2017) Arabic handwritten characters recognition using convolutional neural network. WSEAS Trans Comput Res 5:11–19
19. Balaha HM, Ali HA, Saraya M, Badawy M (2021) A new Arabic handwritten character recognition deep learning system (AHCR-DLS). Neural Comput Appl 33:6325–6367. https://doi.org/10.1007/s00521-020-05397-2
20. Ahmed R, Gogate M, Tahir A, Dashtipour K, Al-tamimi B, Hawalah A, El-Affendi MA, Hussain A (2021) Novel deep convolutional neural network-based contextual recognition of Arabic handwritten scripts. Entropy. https://doi.org/10.3390/e23030340
21. Younis KS (2017) Arabic hand-written character recognition based on deep convolutional neural networks. Jordan J Comput Inf Technol (JJCIT) 3:186–200
22. AlJarrah MN, Zyout MM, Duwairi R (2021) Arabic handwritten characters recognition using convolutional neural network. pp. 182–188 . https://doi.org/10.1109/ICICS52457.2021.9464596
23. Elleuch M, Tagougui N (2015) Arabic handwritten characters recognition using deep belief. Neural Netw. https://doi.org/10.1109/SSD.2015.7348121
24. Elagamy MN, Khalil MM, Ismail E (2023) HACR-MDL: handwritten Arabic character recognition model using deep learning. ISPRS Ann Photogramm Remote Sens Spat Inf Sci 10:123–128
25. Momeni S, BabaAli B (2023) A transformer-based approach for Arabic offline handwritten text recognition
26. Ahmad R, Naz S, Afzal MZ, Rashid SF, Liwicki M, Dengel A (2017) KHATT: a deep learning benchmark on Arabic script. In: 2017 14th IAPR international conference on document analysis and recognition (ICDAR), vol. 07, pp. 10–14 . https://doi.org/10.1109/ICDAR.2017.358
27. Barrere K, Soullard Y, Lemaitre A, Coüasnon B (2022) A light transformer-based architecture for handwritten text recognition. In: Uchida S, Barney E, Eglin V (eds) Document analysis systems. Springer, Cham, pp 275–290
28. Li M, Lv T, Chen J, Cui L, Lu Y, Florencio D, Zhang C, Li Z, Wei F (2022) TrOCR: Transformer-based optical character recognition with pre-trained models
29. Alyahya H, Ismail MMB, Al-Salman A (2020) Deep ensemble neural networks for recognizing isolated Arabic handwritten characters. ACCENTS Trans Image Proc Comput Vis 6:68–79. https://doi.org/10.19101/tipcv.2020.618051
30. Mudhsh M, Almodfer R (2017) Arabic handwritten alphanumeric character recognition using very deep neural. Network. https://doi.org/10.3390/info8030105
31. Al-Taani A, Ahmad S (2021) Recognition of Arabic handwritten characters using residual neural networks Ahmad. Jordan J Comput Inf Technol (JJCIT) 07:192–205
32. Korichi A, Slatnia S, Tagougui N, Zouari R, Kherallah M, Aiadi O (2022) Recognizing Arabic handwritten literal amount using convolutional neural networks. In: International conference on artificial intelligence and its applications, Springer, pp. 153–165
33. Ali AAA, Mallaiah S (2021) Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout. J King Saud Univ Comput Inf Sci 34:3294–3300
34. Wong SC, Gatt A, Stamatescu V, McDonnell MD (2016) Understanding data augmentation for classification: when to warp? CoRR, arxiv.org/abs/1609.08764
35. Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36:193–202. https://doi.org/10.1007/BF00344251
36. Sultana F, Sufian A, Dutta P (2019) Advancements in image classification using convolutional neural network. CoRR, arxiv.org/abs/1905.03288
37. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJ, Bottou L, Weinberger KQ (eds) Advances in neural information processing systems, vol 25. Curran Associates Inc., New York
38. Sun Y, Wang X, Tang X (2015) Deeply learned face representations are sparse, selective, and robust. pp. 2892–2900. https://doi.org/10.1109/CVPR.2015.7298907. http://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298907
39. Yu S, Jia S, Xu C (2017) Convolutional neural networks for hyperspectral image classification. Neurocomputing 219:88–98. https://doi.org/10.1016/j.neucom.2016.09.010
40. Xu X, Liu H (2020) ECG heartbeat classification using convolutional neural networks. IEEE Access 8:8614–8619. https://doi.org/10.1109/ACCESS.2020.2964749
41. Alheraki M, Al-Matham R, Al-Khalifa H (2023) Handwritten Arabic character recognition for children writing using convolutional neural network and stroke identification. Human-Centric Intell Syst 3:147–159. https://doi.org/10.1007/s44230-023-00024-4