Multimedia Tools and Applications (2021) 80:32473–32509
https://doi.org/10.1007/s11042-021-11185-4

Recognizing arabic handwritten characters using deep learning and genetic algorithms

Hossam Magdy Balaha[1] · Hesham Arafat Ali[2] · Esraa Khaled Youssef[3] · Asmaa Elsayed Elsayed[3] · Reem Adel Samak[3] · Mohammed Samy Abdelhaleem[3] · Mohammed Mosa Tolba[3] · Mahmoud Ragab Shehata[3] · Mahmoud Refa'at Mahmoud[3] · Mariam Mahmoud Abdelhameed[3] · Mostafa Mahmoud Mohammed[3]

Received: 20 October 2020 / Revised: 7 May 2021 / Accepted: 24 June 2021 /
Published online: 31 July 2021

Abstract

Automated techniques for Arabic content recognition are at a beginning period contrasted with their partners for the Latin and Chinese contents recognition. There is a bulk of handwritten Arabic archives available in libraries, data centers, historical centers, and workplaces. Digitization of these documents facilitates (1) to preserve and transfer the country's history electronically, (2) to save the physical storage space, (3) to proper handling of the documents, and (4) to enhance the retrieval of information through the Internet and other mediums. Arabic handwritten character recognition (AHCR) systems face several challenges including the unlimited variations in human handwriting and the leakage of large and public databases. In the current study, the segmentation and recognition phases are addressed. The text segmentation challenges and a set of solutions for each challenge are presented. The convolutional neural network (CNN), deep learning approach, is used in the recognition phase. The usage of CNN leads to significant improvements across different machine learning classification algorithms. It facilitates the automatic feature extraction of images. 14 different native CNN architectures are proposed after a set of try-and-error trials. They are trained and tested on the HMBD database that contains 54,115 of the handwritten Arabic characters. Experiments are performed on the native CNN architectures and the best-reported testing accuracy is 91.96%. A transfer learning (TF) and genetic algorithm (GA) approach named "HMB-AHCR-DLGA" is suggested to optimize the training parameters and hyperparameters in the recognition phase. The pre-trained CNN models (VGG16, VGG19, and MobileNetV2) are used in the later approach. Five optimization experiments are performed and the best combinations are reported. The highest reported testing accuracy is 92.88%.

✉ Hossam Magdy Balaha
hossam.m.balaha@mans.edu.eg

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

Keywords Arabic Handwritten Character Recognition (AHCR) · Convolutional Neural Networks (CNN) · Deep Learning (DL) · Genetic Algorithms (GA) · Learning Optimization · Transfer Learning (TF)

1 Introduction

Arabic is a very important international language. The number of Arabic speakers exceeds half a billion (native and non-native) around the world. Arabic is the second most used after the Latin script [67].

Arabic has 28 alphabetical letters, all representing consonants, and are written from right to left [62]. Twenty-two of these letters are those of the Semitic alphabet from which it descended [44]. They are modified only in the letterform. The remaining six letters represent sounds not used in the languages written in the earlier alphabet [100].

The shape of each Arabic letter depends on its position in a word: initial, middle, end, or isolated as shown in Table 1. There are no uppercase nor lowercase distinctions as it is in the Latin characters [125]. Arabic is written cursively both typed and printed [85]. This means, intuitively, the Arabic recognition system needs letter segmentation [43].

Recognizing Arabic writings opens the area of recognizing similar languages that are written with the same script. These similar languages are fourteen including Persian, Urdu, and Pashto [14].

In recent years, there is much interest in the area of handwritten document recognition, especially in Arabic. Among the handwritten and printed documents, the automatic handwritten document recognition is more challenging [35].

Handwritten characters that are written by different persons are not identical and vary in both sizes and shapes. These numerous variations in the writing styles of the individual characters make the recognition process more difficult and challenging [7].

The similarities among distinct character shapes, the overlaps, the ligatures [38], and the interconnections of the neighboring characters [103] further complicate that recognition task.

Little research progress has been achieved compared to what has been done with Latin and Chinese languages [3, 5]. Research in Arabic optical character recognition is still a wide-open area [73].

Arabic texts and their segmentation and recognition challenges [19] can be summarized as follows:

- Arabic text is written typed and printed cursively in blocks of interconnected characters. The cursive nature of the Arabic characters and different forms of Arabic characters makes it more difficult to achieve high accuracy in character classification and recognition.
- One or more blocks can be used to form an Arabic word. This leads to the necessity of an adaptive Arabic text segmentation algorithm.
- Arabic characters in addition to their isolated form can take different shapes depending on their position inside the text: initial, middle, end, or isolated as shown for the letter "Ayn" in Table 1.
- Some characters can be written vertically or horizontally which may lead the character blocks to have more than one baseline or different baselines' locations as shown in Fig. 1. This prevents the simple adaptations of the published baseline detection methods related to the English or Chinese languages.

Table 1 Arabic alphabet characters with different positions

Character	Isolated	Begin	Middle	End	Character	Isolated	Begin	Middle	End
Alif	ا			ـا	Daad	ض	ضـ	ـضـ	ـض
Baa	ب	بـ	ـبـ	ـب	Tah	ط	طـ	ـطـ	ـط
Taa	ت	تـ	ـتـ	ـت	Zah	ظ	ظـ	ـظـ	ـظ
Tha	ث	ثـ	ـثـ	ـث	Ayn	ع	عـ	ـعـ	ـع
Jim	ج	جـ	ـجـ	ـج	Ghayn	غ	غـ	ـغـ	ـغ
Haa	ح	حـ	ـحـ	ـح	Faa	ف	فـ	ـفـ	ـف
Khaa	خ	خـ	ـخـ	ـخ	Qaf	ق	قـ	ـقـ	ـق
Dal	د			ـد	Kaaf	ك	كـ	ـكـ	ـك
The	ذ			ـذ	Laam	ل	لـ	ـلـ	ـل
Raa	ر			ـر	Meem	م	مـ	ـمـ	ـم
Zai	ز			ـز	Nun	ن	نـ	ـنـ	ـن
Sin	س	سـ	ـسـ	ـس	Heh	ه	هـ	ـهـ	ـه
Shen	ش	شـ	ـشـ	ـش	Waw	و		ـو	
Saad	ص	صـ	ـصـ	ـص	Yaa	ي	يـ	ـيـ	ـي

- Arabic has diacritical marks that must be dealt with [55].
- Arabic uses different types of external objects such as dots (single, double, or triple), "Hamza", and "Madda" [39].
- The same character can have substantial length variation in each different text as shown in Fig. 2.
- Handwritten characters vary in both sizes and shapes and some combinations of the Arabic characters have unique forms.
- There are not enough public and available Arabic handwritten text and characters datasets to work on compared to the Latin language.

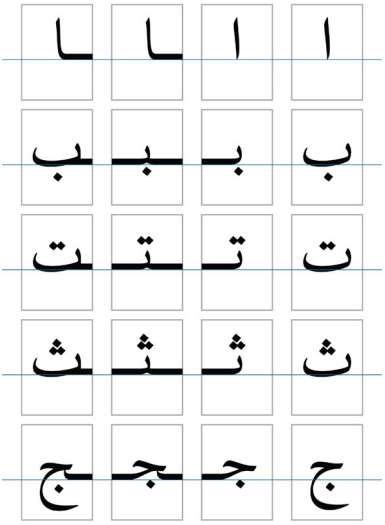

Fig. 1 Four shapes of five arabic letters "Alif", "Baa", "Taa", "Tha", and "Jim" with baselines

Fig. 2 Three different shapes of the word "Mohammed (محمد)"

Pattern recognition is a branch that covers various fields such as face recognition, fingerprint recognition, image recognition, citywide traffic crowd flows, character recognition, and numerals recognition [10, 11, 61].

Arabic handwritten character recognition (AHCR) system is an intelligent system that can classify and recognize handwritten characters similar to humans [94]. The input to the AHCR system is the text required to be recognized and the output is the corresponding recognized text in the digital form. Figure 3 shows sample input and output of an AHCR system.

There have been different machine learning methods and techniques that are used for offline handwritten characters classification and recognition [8] such as support vector machines [16, 110] and K-nearest neighbor [66]. The traditional machine learning methods extract the features manually using feature engineering [63]. After that, classification algorithms are used to classify the characters based on the extracted features.

Deep learning (DL) is a trending research machine learning branch [46] that includes different algorithms and architectures such as: Deep Neural Networks [30], Convolutional Neural Networks (CNNs) [70], Deep Belief Networks [56], Autoencoders [21], and Recurrent Neural Networks [108]. They are capable of extracting the features automatically [78].

Deep neural networks consist of an input layer, multiple non-linear hidden layers, and an output layer. The number of connections and trainable parameters can become very large and the network does not fit in the device memory especially with images and videos [83]. The deep neural network needs also a very large dataset to avoid overfitting [31].

A CNN has a comparatively smaller set of parameters (i.e. weights) and easier to train [77]. It is a multi-layer feed-forward neural network that extracts salient features and properties from the input data automatically. It uses the neural network back-propagation algorithm [57] in its training.

A CNN adds the new dimension for image classification systems and recognizing visual patterns directly from pixel images with minimal pre-processing [4]. Besides, It automatically provides some degree of translation invariance [29]. It can learn from high-dimensional complex inputs, non-linear mappings from a very large number of records [80].

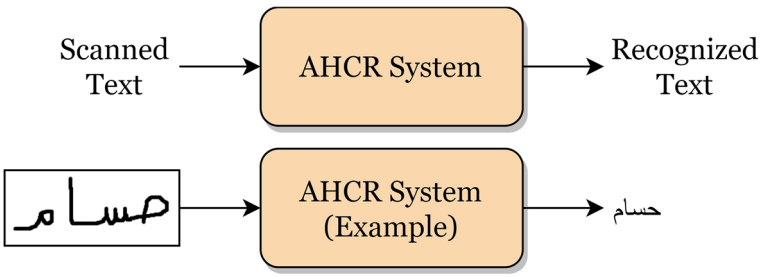

Fig. 3 Sample input and output of the AHCR system

Another major advantage of CNN is the usage of the shared weights in the convolutional layers. This means that the same filter is used for each input in the layer [76]. The shared weights reduce the parameter numbers and improve the overall performance [18]. Recently, CNN is found more efficient for handwritten character recognition due to its distinct features [112].

Instead of learning the CNN from scratch, the transfer learning (TF) approach can be used. Transfer learning is a machine learning approach that focuses on the gained knowledge and experience storage while working on a task or a problem [93]. It can be applied to different related problems.

This TF approach is engaged in deep learning by working on these stored pre-trained models as the starting points. This allows rapid progress and performance improvement [79].

The pre-trained model can be achieved by two common paths (1) choosing a related predictive modeling task with an abundance of data where there are relationships in the input and output data or (2) selecting a pre-trained source model from the available ones released by companies, research institutions, universities, etc. [137].

There are many available pre-trained CNN models such as VGG16 [113], ResNet [52], MobileNet [58], Xception [27], NASNet [138], and DenseNet [59]. These pre-trained models were trained on the ImageNet [33]. It is an very large scale hierarchical image database (over 14 million images). It was designed for visual object recognition software research and computer vision tasks [102].

Selecting between the different hyperparameters (e.g. batch size, parameters initializers, and activation functions) used in the training of CNN models can lead to a non-deterministic polynomial-time hard (NP-hard) problem [49].

Determining the optimal set of training hyperparameters to achieve the maximum performance can be considered as an optimization problem that can be solved using soft computing algorithms such as genetic algorithms (GAs) and fuzzy logic systems [99].

Hence, the current study presents an AHCR system for text segmentation and recognition. The recognition uses CNNs and TF for the parameters optimizations and uses GA for the hyperparameters optimizations.

The main contributions of the current study can be summarized as follows:

- Discussing the Arabic handwritten character recognition system.
- Presenting the text segmentation challenges and suggesting a set of solutions for each challenge.
- Suggesting different native CNN architectures for the problem of AHCR.
- Applying transfer learning for the AHCR problem using different pre-trained CNN models.
- Proposing a TF and genetic algorithm approach, HMB-AHCR-DLGA, to optimize the AHCR training parameters and hyperparameters.
- Performing different experiments and reporting the best architectures using different performance metrics. Reporting the effectiveness of the best ones.

The rest of the paper is organized as follows: Section 2 presents the related works. Section 3 presents the problem formulation, discusses the text segmentation task, and outlines the different challenges with suggested solutions, presents the classification task using native and transfer learning approaches, and proposes the TF and GA approach to optimize the AHCR training parameters and hyperparameters. Section 4 presents the experiments, results, and discussions, and finally, Section 5 concludes the paper and presents future work.

2 Related work

According to the available and public publications, many researchers addressed the classification and recognition of letters including Arabic. On the other hand, there are a smaller number of methods and approaches used for the Arabic language that had been explored for recognizing the individual characters. Hence, Arabic optical character recognition is less common compared to English, French, and Chinese [12].

Various methods and approaches were presented and high recognition rates were achieved and reported for the handwritten recognition of English [18, 132], Chinese [51, 126, 131, 135, 136], Hangul [40], Malayalam [87], Devanagari [2, 114], and Telugu [116] using CNN.

Shams et al. [111] presented an algorithm for recognizing Arabic characters based on deep CNN and support vector machine. They used the correct classification rate (CRR) and error classification rate (ECR). Their proposed system used a K-means clustering algorithm to check the Arabic characters' similarity. They proposed a sample dataset. They applied 840 tested images from 3 persons where each one wrote each character 10 times. Their reported rates were 95.07% CRR and 4.93% ECR.

Altwaijry et al. [13] presented a new dataset (named "Hijja") of Arabic letters written that were compiled by children aged 7 to 12. It contained 47,434 characters written by 591 participants. They proposed an automatic handwriting recognition architecture based on CNN. They trained their model on "Hijja" and "AHCD" [36] datasets. They reported accuracies of 97% and 88% on the "AHCD" and "Hijja" datasets respectively.

Kef et al. [68] presented a Fuzzy ARTMAP [48] neural network. The fuzzy ARTMAP neural network is an incremental supervised learning classifier. Five Fuzzy ARTMAP neural networks were used to classify characters using the "IFN/ENIT" database [95]. The "IFN/ENIT" database contains 3,840 handwritten character images (2,304 for training and 1,536 for testing). The reported recognition rate was 93.80%.

ElAdel et al. [37] presented a Neural Network architecture based on the Fast Wavelet Transform, Multi-Resolution Analysis, and Adaboost algorithm. The learning and testing of the Arabic handwriting character classification system were conducted using the "IESK-arDB" [41] dataset which includes 6,000 segmented characters. The classification rate for the different groups of characters was 93.92%.

Elleuch et al. [40] introduced an Arabic handwritten character's recognition using Deep Belief Neural Networks. The approach was tested on the "HACDB" database [74]. The "HACDB" database contains 6,600 shapes of handwritten characters written by 50 persons. The dataset is divided into a training set of 5,280 images and a test set of 1,320 images. The classification result was promising with an error rate of 2.1% (97.90% accuracy).

Lawgali et al. [75] proposed a framework for the recognition of handwritten Arabic words based on segmentation. It involved two phases: training and testing phases. In the training phase, They trained Arabic handwritten characters to be recognized in the next phase, while in the testing phase, they segmented words into characters for recognition. They used the IFN/ENIT database [95]. Their reported accuracy was 90.73% based on their used segmentation method.

Mozaffari et al. [87] proposed a method for the isolated Farsi/Arabic handwritten numerals recognition. They used quad tree-based fractal representation and iterated function systems. The reported recognition rate was 92.60% and was obtained on their numeral database. Their database contained 480 samples per digit and was gathered from nearly 200 people. They had different ages and educational backgrounds.

Al-Shaher et al. [6] presented training point distribution models using the expectation-maximization algorithm. They designed an optical character recognition system and trained their system using eight Arabic letters. They used a small database tested on 7-character classes. They used 100 samples of each character and reported a successful recognition rate of 98.30%.

Amin et al. [15] used conventional machine learning methods. They used rule-based stroke types to extract the relationships automatically. Their inductive learning program generated first-order Horn clauses for characters. The system was tested on their sample handwritten characters that were compiled from several individuals. It contained 120 characters of 40 samples (30 for training and 10 for testing). Their reported accuracy was 86.65%.

Abuhaiba et al. [1] presented a text recognition system that was capable of recognizing offline handwritten Arabic cursive text. Their system was tested against the handwritings of 20 subjects. Their reported accuracies were 55.40% and 51.10% for sub-word and character recognition rates respectively.

Table 2 summarizes the presented related works and studies. The table is ordered from the latest to the oldest.

3 Problem formulation and suggested approach

Arabic handwritten character recognition (AHCR) system accepts the input scanned physical image that contains the Arabic text and outputs the corresponding digital form of the text. The AHCR system consists of (1) image acquisition phase, (2) image pre-processing phase, (3) image segmentation phase, (4) classification phase, and (5) output phase [9] as shown in Fig. 4.

The image acquisition phase converts the image from the physical form into a digital one. There are different image acquisition tools and equipment such as scanners and cameras.

The image pre-processing phase targets to refine the image globally and improves its quality for the next phases [42, 117]. There are many pre-processing techniques such as noise removal [86], grayscale conversion, and slant and skew correction [26, 115].

The image segmentation phase performs three internal sub-phases. They are (1) page-to-lines segmentation, (2) line-to-words segmentation, and (3) word-to-characters segmentation [32, 92].

An Arabic handwritten recognition system can depend only on the first two sub-phases if the classifier (in the fourth phase) works on the words instead of the characters.

The classification phase accepts each character's sub-image and outputs the corresponding digital format of that character by passing it through the trained model.

Finally, the output phase presents the result to the user by merging the characters into words, words into lines, and lines into a page concerning their order from the image segmentation phase.

3.1 Pre-processing phase

The pre-processing phase is essential in enhancing the quality of the images (in our case). They are numerous techniques that can be used including grayscale conversion, binary conversion, noise removal, slant and skew correction, and histogram equalization.

Table 2 Character recognition systems related works and studies summarization

Author(s)	Year	Approach	Dataset	Accuracy
Shams et al. [111]	2020	Convolutional Neural Network (CNN) architecture for training and K-means clustering algorithm to check the Arabic characters similarity	Their proposed dataset. It contained 840 tested images from 3 persons where each one wrote each character 10 times.	95.07%
Altwaijry et al. [13]	2020	Convolutional Neural Network (CNN) architecture.	Their proposed dataset (Hijja). It contained 47,434 characters written by 591 participants. They used also the AHCD [36] dataset.	97.00% (AHCD) and 88.00% (Hijja)
Kef et al. [68]	2016	Five Fuzzy ARTMAP (incremental supervised learning classifier) neural networks were used to classify characters.	The IFN/ENIT database [95].	93.80%
ElAdel et al. [37]	2015	Neural Network architecture based on Fast Wavelet Transform, Multi-Resolution Analysis, and Adaboost algorithm.	The "IESK-arDB" dataset [41].	93.92%
Elleuch et al. [40]	2015	A system using Deep Belief Neural Networks.	The "HACDB" database [74].	97.90%
Lawgali et al. [75]	2014	The discrete cosine transformation to obtain the features of the character. They used them to identify the shape of the character using neural networks.	IFN/ENIT database [95].	90.73%
Mozaffari et al. [87]	2005	Quadtree, fractal encoding, nearest neighbor, and minimum Euclidean distance.	Their proposed dataset. It contained 480 samples per digit and was gathered from nearly 200 people. They had different ages and educational backgrounds.	92.60%
Al-Shaher et al. [6]	2003	Training point distribution models using the expectation-maximization algorithm.	Their proposed dataset. It consisted of 7-character classes using 100 samples of each character.	98.30%

Table 2 (continued)

Author(s)	Year	Approach	Dataset	Accuracy
Amin et al. [15]	2003	Rule-based stroke types and relationships were extracted automatically. The inductive learning program generated first-order Horn clauses for characters.	Their proposed dataset. It contained 120 characters of 40 samples (30 for training and 10 for testing).	86.65%
Abuhaiba et al. [1]	1998	Fuzzy sequential machine character recognition.	Their proposed dataset. It contained 13 pages for training and 20 pages for testing.	55.40% (sub-word) and 51.10% (character)

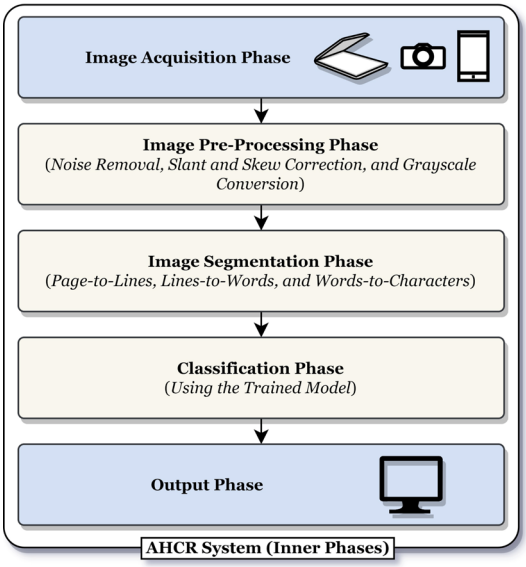

Fig. 4 The phases of an AHCR system

The grayscale conversion is a technique that converts from the color format into the grayscale format. There are different methods such as average and weighted method (i.e. luminosity method) [105].

The binary conversion is a technique that converts the image into a monochromatic black and white image (i.e. bi-level or two-level) [72].

The slant and skew correction technique is used to deskew the image to return it to its original orientation. Several studies addressed different solutions and reviews that target that problem [17, 65, 69].

The noise removal (i.e. image denoising) technique targets the removal or reduction of the unrequired elements from the image. There are numerous methods such as linear, min, max, median, Wiener, Gaussian, and Guided filtering [130].

3.2 Text segmentation phase

The text segmentation phase accepts a textual image and extracts the characters from it. The text segmentation phase passes through three sub-phases: (1) text-to-lines segmentation, (2) lines-to-words segmentation, and finally, (3) words-to-characters segmentation. The output of this phase should be the Arabic letters that should be passed after that to the recognition phase.

3.2.1 Segmentation challenges

There are different challenges associated with the text segmentation phase. Some of them can be handled and refined in the pre-processing phase such as noise removal and the others require additional processing [90]. They can be presented as follows:

- **Differing Acquisition Methods**: The image file format is not specified and the phase is required to be able to handle the different image formats from different sources such as scanners, digital cameras, and mobile cameras.
- **Different Encodings**: The phase should not be affected by the information and lost details by the different lossy and lossless encoding methods [124].

- **Varying Lighting**: The image acquisition environment can vary in lighting conditions.
- **Varying Orientations**: The text can be skewed or rotated and the text segmentation phase should detect and correct that.
- **Varying Text Color**: The text required to be detected and segmented can include multiple colors in the same image.
- **Varying Text Font**: The text is not specified to be computer printed or handwritten and the text segmentation phase should handle both of them.
- **Varying Text Size**: The text size can vary in the same image.
- **Varying Relative Size**: The relative size of the text to the image size is not specified and the module should be able to handle any variations.
- **Multiple Objects on the Same Image**: Non-textual element can be part of the textual image.
- **Non-Clear Background**: Text can be written on a writing line, not on a blank white paper.
- **Non-Uniform Letter Box**: Since handwritten text segmentation is targeting the bounding box, the letter can be in a non-rectangular form and can overlap with other letters boxes [128].
- **Limited Processing Time**: The text segmentation phase is a part of a bigger system or a framework. Hence, it should be functioning in a near real-time manner.

The main challenge in the current study is segmenting Arabic connected cursive text as the rest of the challenges can be avoided or were solved previously in the previous studies.

3.2.2 Segmentation suggested solution

A standard optical character recognition module design, a multi-stage module, is followed. The task is divided into multiple sub-phases (i.e. steps). It makes the design simpler, however, the errors accumulate over the steps. The steps in the suggested solution are as follows:

- **Step 1: Page Layout Analysis**: This step detects the text and non-text regions of the textual image.
- **Step 2: Image Normalization**: To make processing time acceptable, the image is normalized by scaling it to an acceptable size in the current step.
- **Step 3: Binarization and Noise Removal**: This step should remove the non-textual parts from the textual image and making the text in a solid black color.
- **Step 4: Skew Correction and Rotation**: The image skew is detected and corrected by rotating it.
- **Step 5: Finding Writing Lines**: This step targets the baselines on the different writing lines.
- **Step 6: Segmenting Connected Components**: The connected components can be a character or multiple characters connected together (i.e. sub-words).
- **Step 7: Generating Multiple Possible Final Segmentations**: These segmentations will be passed to the recognition module to be evaluated.
- **Step 8: Final Segmentation Choice**: Based on the score returned from the recognition module for each segmentation, this step will decide which segmentation is the best to be the final one.

Table 3 summarizes the challenges with their solutions.

Table 3 Segmentation challenges and solutions summarization

Challenge	Solution
Differing Acquisition Methods	Using pre-existing libraries such as OpenCV [25] to read the image as an array of colored pixels regardless of how it was represented.
Different Encodings	Encoding artifacts are removed in the noise removal step.
Varying Lighting	Lighting variation is removed in the binarization step.
Varying Orientation	Fixed in the skew correction and rotation step.
Varying Text Color	Removed in the binarization step.
Varying Text Font	Handled by the feedback mechanism step.
Varying Text Size	Handled in segmenting the connected components step.
Varying Relative Size	Handled in finding the writing lines step.
Multiple Objects on the Same Image	Handled in page layout analysis step.
Non-Clear Background	Handled in the binarization and noise removal step.
Non-Uniform Letter Box	Handled by the feedback mechanism step.
Limited Processing Time	Handled by the normalization step.

3.2.3 Main challenge discussion

This is mainly handled in steps 7 and 8. In step 7, the segments are generated based on the features extracted from the connected component. A set of rules are then used to form these segments based on the features. These features can be structural features such as loops, endpoints, X-cross points, and T-cross points, or statistical features such as moments and Fourier descriptors [54].

In step 8, the tag scores are used to determine how likely the segmentation is valid. A suggested function would be based on Bayes' rule [121] by treating each score as probability and trying to find the minimum noise probability.

The following example summarizes the discussion (the numbers and tags here are manually generated to give an illustration and not the actual results). For the 7th step, Fig. 5 shows one possible connected component box and Fig. 6 shows multiple possible generated segmentations. The multiple segmentations are passed to the recognition module and the module assigns tags and numbers corresponding to each tag as shown in Table 4.

For the 8th step, the goal of this function is to maximize the letter score and minimize the noise score in the overall segmentation. A function is designed to determine the best segmentation based on the tags and scores.

Some suggested functions: (1) **Minimizing the Sum of Noise**: The sum of noise score is calculated for each possible segmentation and the one having the least noise sum is selected to be the accepted segmentation element and (2) **Minimizing the Root of the Mean Squares Summation (RMSS)** as shown in (1). They require no information other than the

Fig. 5 One possible connected component box for the word "مثلث"

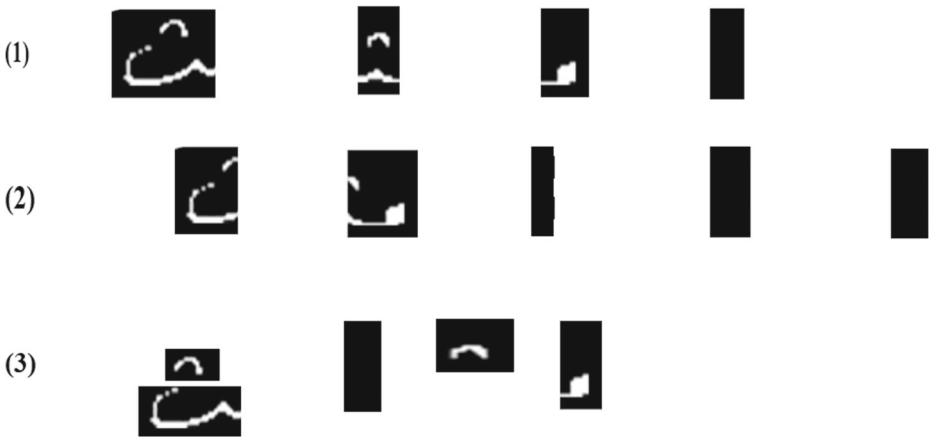

Fig. 6 Multiple possible segmentations for the word "مثلث"

Table 4 Example on the segmentation tags and their corresponding numbers

Character tag	Corresponding numbers
	Tha (6), Faa (3) and Taa (1)
	Laam (5), Raa (3), Alif (15) and Noise (5)
	Tha (7), Taa (2) and Nun (1)
	Meem (87), Waw (1) and Noise (3)
	Tha (4), Laam (2), Kaaf (2) and Noise (2)
	Noise (7), Taa (2) and Kaaf (1)
	Meem (6), Noise (2), Faa (15) and Waw (5)
	Raa (8) and Zai (2)

segmentation and the noise scores.

$$RMSS = \sqrt{\sum \left(\frac{NoiseScore}{Number of Letter Boxes} \right)^2} \tag{1}$$

3.3 Classification phase

As mentioned earlier, the classification phase accepts each character's sub-image and outputs the corresponding digital format of that character by passing it through the trained model. The model can be trained from scratch or using the transfer learning approach with the help of the pre-trained CNN models.

This subsection is divided into three inner subsections: (1) designing CNN models from scratch, (2) using pre-trained models (transfer learning approach), and (3) proposing a TF and genetic algorithm (GA) optimization approach for the best training parameters and hyperparameters selection.

3.3.1 Designing CNN models

A Convolutional Neural Network (CNN) is a multi-layer neural network designed to analyze the visual inputs and perform tasks. The input data in our case is the images X as shown in (2).

$$X = [x_1, x_2, ..., x_j, ..., x_n] \tag{2}$$

where X is the whole dataset, x_j is a dataset entity (i.e. element) at a position j, and n is the number of elements in X. Formally, the input to a convolutional layer is an image with a shape shown in (3).

$$shape(x_j) = M \times M \times C \tag{3}$$

where M is the size of the image (i.e. width and height) and C is the number of channels per pixel. For a grayscale image, it has one channel and hence $C = 1$ but Red-Green-Blue (RGB) images have three channels and hence $C = 3$.

CNN is composed of several kinds of layers:

Convolutional layers The convolutional layer has K filters (i.e. kernels) each filter has a shape that can be computed from (4).

$$shape(ConvLayer) = N \times N \times R \tag{4}$$

where N is the size of the filter (i.e. width and height) and R is the number of channels. R should be equal to or less than C. Convolutional layer creates K feature maps of size $M - N + 1$. They are used to predict the class probabilities for each feature by applying a filter that scans the whole image, a few pixels at a time.

Figure 7 illustrates the convolution process on an input image of size $M \times M$ with a kernel of size $N \times N$. If the input is 32×32 and the kernel is 5×5, then the output of the convolution is 28×28.

Pooling layers Each feature map is then pooled typically with mean-, min- or max-pooling over a $Q \times Q$ where Q is a range between 2 to 5 for large inputs.

Figure 8 illustrates the pooling process on the feature map that produces a new feature map of size $(M - N + 1)/2$. If the input is 28×28, then the output of the pooling layer is 14×14.

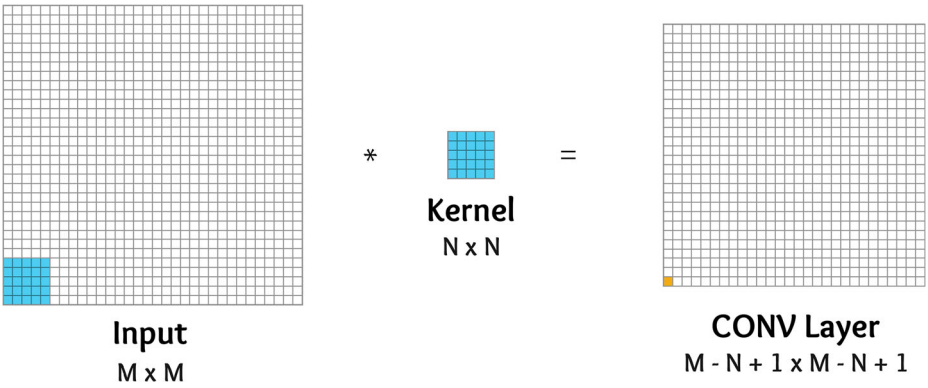

Fig. 7 Illustration of the convolution process on an input image of size $M \times M$ with a kernel of size $N \times N$

Pooling layers are used to scale down the amount of information the convolutional layer generates for each feature and maintains the most essential information. Notice that, the process of the convolutional and pooling layers usually are repeated several times [106].

Fully-connected layers After the convolutional and pooling layers, one or more fully-connected layers exist(s). The high-level reasoning in the neural network is done via the fully-connected layers.

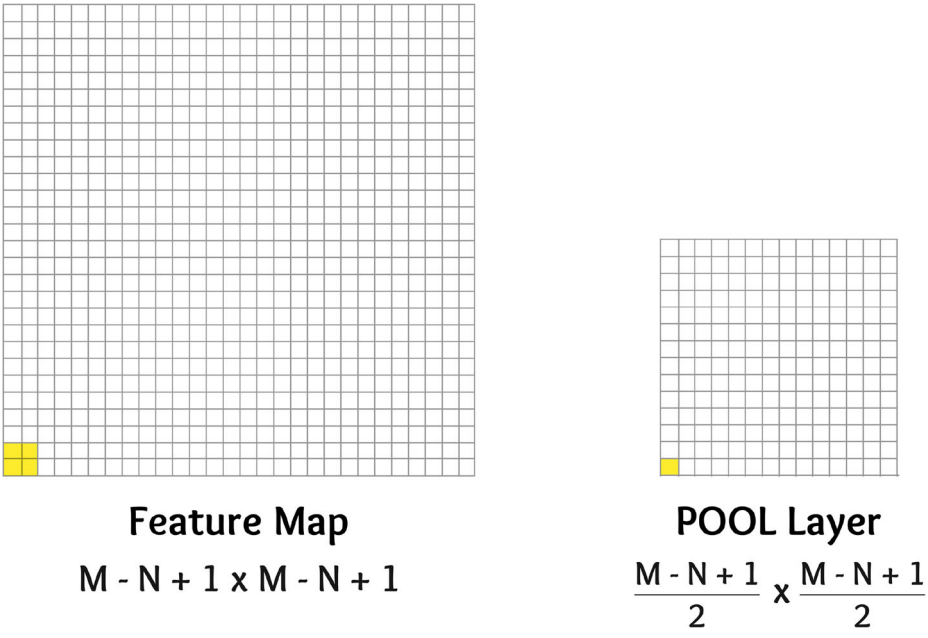

Fig. 8 Illustration of the pooling process on the feature map

A fully-connected layer accepts all neurons from the previous layer and connects them to every single neuron it has. Fully-connected layers are not spatially located anymore, so there can be no convolutional layers after a fully connected layer.

Activation functions Activation functions are non-linear transformations used to determine whether a neuron should be activated or not. There are several activation functions such as Sigmoid, ReLU (Rectified Linear Unit), and Leaky ReLU [91].

The activation function that is used in the current study for the hidden layers of the designed CNN models is ReLU [89].

ReLU is a piece-wise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for neural network types because the architecture that uses it is easier to train and often achieves better performance than others [50].

The SoftMax classifier function is a generalization of the binary form of Logistic regression [34]. It is commonly used for multi-class classification problems. The mapping function is defined such that it takes an input dataset X and maps them to the output class labels via a simple linear dot product of the data X and weights W.

Learning weights optimization Optimization algorithms and techniques help us to minimize (or maximize) an objective function (i.e. the error function). An error function is simply a mathematical function that is dependent on the model's internal learnable parameters. It is used to compare the target values of Y and model predictions.

The gradient descent algorithm [101] updates the parameters to minimize the error function. It takes small steps in the direction of the negative gradient of the loss function as shown in (5).

$$P_{i+1} = P_i - \alpha \times \nabla \times E(P_i) \tag{5}$$

where i refers to the iteration number, ∇ is the learning rate, E is the error function, P is the parameter vector and P_i is the loss function. The gradient of the loss function, $\nabla \times E(P_i)$, is evaluated using the entire training dataset. The standard gradient descent algorithm uses the entire dataset at once.

The learning rate is a hyperparameter that controls how much we are adjusting the trainable weights of the network concerning the calculated loss gradient. The lower the learning rate value, the slower we move along the downward slope and the more time it takes.

Adam (Adaptive Moment Optimization Algorithm) [71] combines the heuristics of both the Momentum [129] and RMSProp [22] optimization algorithms. They take contrasting approaches. The Momentum optimizer accelerates the search in the direction of the minima while the RMSProp optimizer impedes the search in the direction of oscillations.

There are other weights optimizers such as Nadam [119], AdaGrad [88, 122], AdaDelta [133], AdaMax [60], and FTRL [82].

The authors prepared 14 CNN architectures after try-and-error trials where 3 of them achieved the best results as shown later in the experiments and results section. The 14 CNN architectures are labeled from 1 to 14, so that, the first one is CNN-1, the second one is CNN-2, and so on.

Figures 9, 10, and 11 show the top-3 suggested CNN architectures CNN-5, CNN-12, and CNN-11 respectively for Arabic handwritten character recognition. The layers' sizes and types are labeled on the figures.

All of the suggested CNN architectures have the last output fully-connected layer with 115 classes. This number refers to the number of classes of the used dataset. The

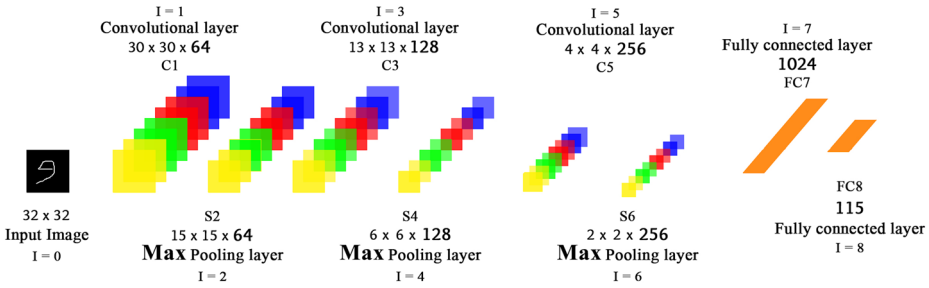

Fig. 9 A suggested CNN architecture, CNN-5, to classify the arabic handwritten characters

used dataset is described later in the experiments and results section. The used activation functions in the hidden and output layers are ReLU and SoftMax respectively.

Figure 9, CNN-5, is described as follows: $INPUT \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow FC \rightarrow RELU \rightarrow FC$. The input layer accepts images with a size of 32×32. The second layer, $C1$, is a convolutional layer with a size of 30×30 and 64 parallel filters. The third layer, $S2$, is a max-pooling layer with a size of 15×15 and 64 parallel kernels.

The fourth layer, $C3$, is a convolutional layer with a size of 13×13 and 128 parallel filters. The fifth layer, $S4$, is a max-pooling layer with a size of 6×6 and 128 parallel kernels. The sixth layer, $C5$, is a convolutional layer with a size of 4×4 and 256 parallel filters.

The seventh layer, $S6$, is a max-pooling layer with a size of 2×2 and 256 parallel kernels. The eighth layer, $FC7$, is a fully-connected layer with a number of neurons of 1024. The last layer, $FC8$, is another fully-connected layer with a number of neurons of 115.

Figure 10, CNN-12, is described as follows: $INPUT \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow FC \rightarrow RELU \rightarrow FC \rightarrow RELU \rightarrow FC$. The input layer accepts images with a size of 32×32. The second layer, $C1$, is a convolutional layer with a size of 30×30 and 64 parallel filters. The third layer, $S2$, is a max-pooling layer with a size of 15×15 and 64 parallel kernels.

The fourth layer, $C3$, is a convolutional layer with a size of 13×13 and 128 parallel filters. The fifth layer, $S4$, is a max-pooling layer with a size of 6×6 and 128 parallel kernels. The sixth layer, $C5$, is a convolutional layer with a size of 4×4 and 256 parallel

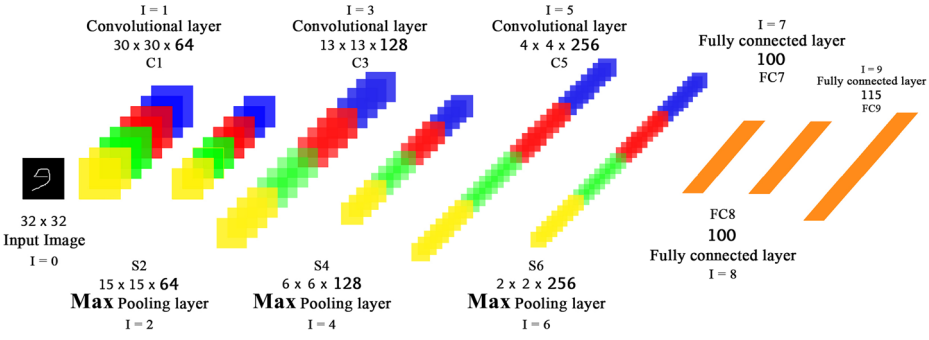

Fig. 10 A suggested CNN architecture, CNN-12, to classify the arabic handwritten characters

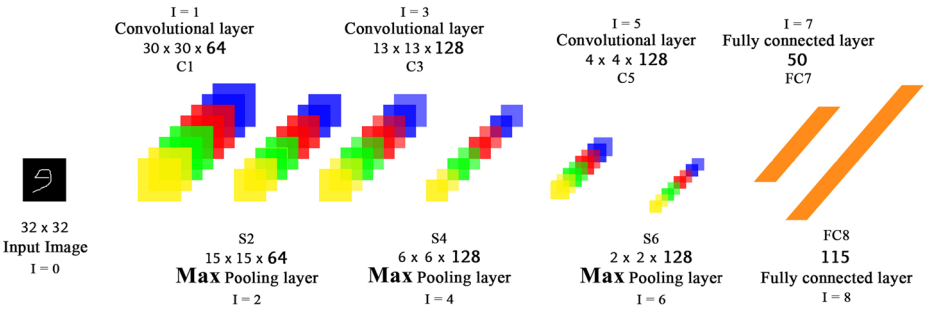

Fig. 11 A suggested CNN architecture, CNN-11, to classify the arabic handwritten characters

filters. The seventh layer, $S6$, is a max-pooling layer with a size of 2×2 and 256 parallel kernels.

The eighth layer, $FC7$, is a fully-connected layer with a number of neurons of 100. The ninth layer, $FC8$, is another fully-connected layer with a number of neurons of 100. The last layer, $FC9$, is also another fully-connected layer with a number of neurons of 115.

Figure 11, CNN-11, is described as follows: $INPUT \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow CONV \rightarrow RELU \rightarrow POOL \rightarrow FC \rightarrow RELU \rightarrow FC$. The input layer accepts images with a size of 32×32. The second layer, $C1$, is a convolutional layer with a size of 30×30 and 64 parallel filters. The third layer, $S2$, is a max-pooling layer with a size of 15×15 and 64 parallel kernels.

The fourth layer, $C3$, is a convolutional layer with a size of 13×13 and 128 parallel filters. The fifth layer, $S4$, is a max-pooling layer with a size of 6×6 and 128 parallel kernels. The sixth layer, $C5$, is a convolutional layer with a size of 4×4 and 128 parallel filters. The seventh layer, $S6$, is a max-pooling layer with a size of 2×2 and 128 parallel kernels.

The eighth layer, $FC7$, is a fully-connected layer with a number of neurons of 50. The last layer, $FC8$, is also another fully-connected layer with a number of neurons of 115.

3.3.2 Transfer learning approach

VGG16 [113], VGG19 [113], and MobileNetV2 [104] pre-trained CNN models are used in the current study as the base models. The ImageNet [33] weights are used with them to initiate their weights.

Each pre-trained model consists of a set of layers internally and we can control which layers to train and update their weights. For example, freezing all the pre-trained model layers unless the last two layers. In the training (in the current study), the last two layers' weights will be updated while the rest of the layers will remain with their initial weights.

By decreasing the number of trainable layers, the number of trainable weights will decrease and hence the training time. This may lead to a noticeable decrease in the accuracy and hence selecting a suitable learning percent for the pre-trained model layers is necessary.

A global average pooling layer [28], dropout layer [118, 127], and a fully-connected layer are added after the pre-trained model's last layer. Similar to max-pooling layers, the global average pooling layers are used to reduce the spatial dimensions (i.e. dimensionality reduction) but in an extreme manner where the feature maps layer with dimensions $M - N + 1 \times M - N + 1 \times R$ is reduced in size to have dimensions of $1 \times 1 \times R$ as shown in Fig. 12.

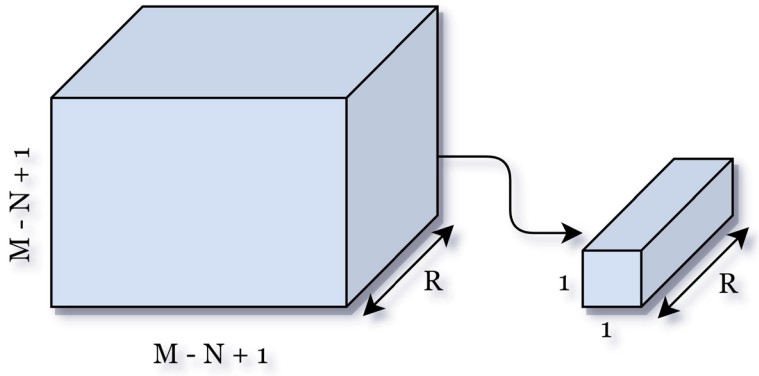

Fig. 12 Illustration of the global average pooling process on the feature maps

Dropout layer Dropout is a regularization technique that refers to dropping some of the neurons of the layer that precedes this layer as shown in Fig. 13. Dropout layers are important in training the CNNs as they prevent overfitting [134] on the training data.

Selecting a suitable dropout ratio is necessary as selecting a very high ratio will decrease the number of trainable neurons and may lead to a decrease in accuracy.

3.3.3 Proposed TF and GA parameters and hyperparameters optimization approach (HMB-AHCR-DLGA)

From the previous subsections, different DL training hyperparameters are required to be optimized. They are the (1) DL weights optimizers, (2) dropout ratios, (3) batch sizes, and (4) transfer learning model learning ratios in the current study.

Depending on the grid search [53] only, it will lead to $a \times b \times c \times d$ tries where a, b, c, and d are the numbers of DL weights optimizers, dropout ratios, batch sizes, and

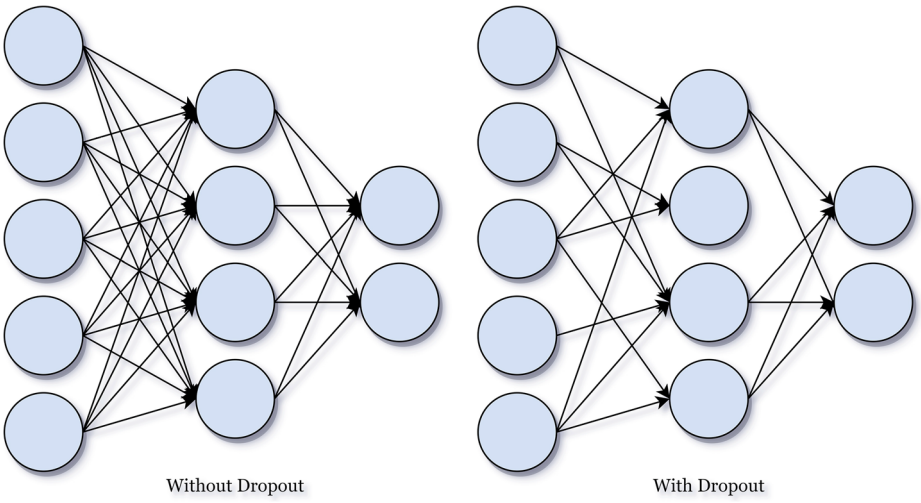

Fig. 13 Illustration of the dropout process

TF model learning ratios respectively. Hence, a suitable optimization approach for these hyperparameters is required to be applied.

A genetic algorithm (GA) [123] approach is used to solve this issue and to achieve the best combinations faster than the grid search or native searching approach.

GA has different important features [84] such as (1) it combines different solutions to boost the best one, (2) it is used to solve many problems effectively, (3) simpler to implement and deploy, and (4) requires fewer computations compared to other state-of-the-art approaches [45].

The proposed genetic algorithm approach named HMB-AHCR-DLGA is described in detail in this subsection. Initially, all chromosomes are initiated randomly concerning the given ranges (i.e. DL weights optimizers, dropout ratios, batch sizes, and TF model learning ratios). They are constructed as a matrix where each row is a chromosome.

A GA is applied to the whole population. The algorithm steps include the objective function calculation, sorting, selection, crossover, and mutation. After a set of iterations, the final best chromosome that includes the combination with the highest objective function value is achieved upon completion. The steps are described comprehensively in the following paragraphs.

Initial poulation The initial population is created as a matrix where each row represents a chromosome. The size of the population is initiated as N_p. Each chromosome (i.e. row) has a set of genes (i.e. columns). The size of each chromosome is set to 4.

They are (1) the first column is for a DL weights optimizer, (2) the second column is for a dropout ratio, (3) the third column is for batch size, and (4) the fourth column is for a TF model learning ratio as shown in Fig. 14.

The used DL weights optimizers are Adam, Nadam [119], AdaGrad [88, 122], AdaDelta [133], AdaMax [60], RMSProp, SGD, and FTRL [82]. The TF model learning ratio is ranged from 0% to 60% with a step of 5% and hence there are 13 learning ratios.

The batch size is ranged from 32 to 128 with a step of 32 and hence there are 3 sizes. The dropout ratio is ranged from 0% to 60% with a step of 1% and hence there are 61 dropout ratios.

Objective function Each chromosome is applied to the objective function to determine the corresponding score. The more the value, the better the chromosome. The used objective function is a multi-objective function consisting of the deep learning performance metrics [64].

They are accuracy, loss, F1-score, recall, the area under curve (AUC), and precision. The more the accuracy, F1-score, recall, and precision, the better the deep learning model. The lower the loss, the better the deep learning model. To solve the loss value issue to maximize it, the value of $1/loss$ is computed instead of the $loss$ value.

The used objective function is designed in a way to maximize and integrate the objectives. To solve the multi-objective optimization problem, the weighted sum approach (WSA) [81] is used. It multiplies each objective by a weight w and finds the sum of them as shown in (6).

$$\begin{aligned} Value = {} & w_1 \times Accuracy + w_2 \times \frac{1}{Loss} + \\ & w_3 \times Recall + w_4 \times F1 + \\ & w_5 \times Precision + w_6 \times AUC \end{aligned} \tag{6}$$

where w_1, w_2, w_3, w_4, w_5, and w_6 are the weight values and their summation equals 1. The accuracy is most intuitive performance metric [23] and hence its weight w_1 is set to be the

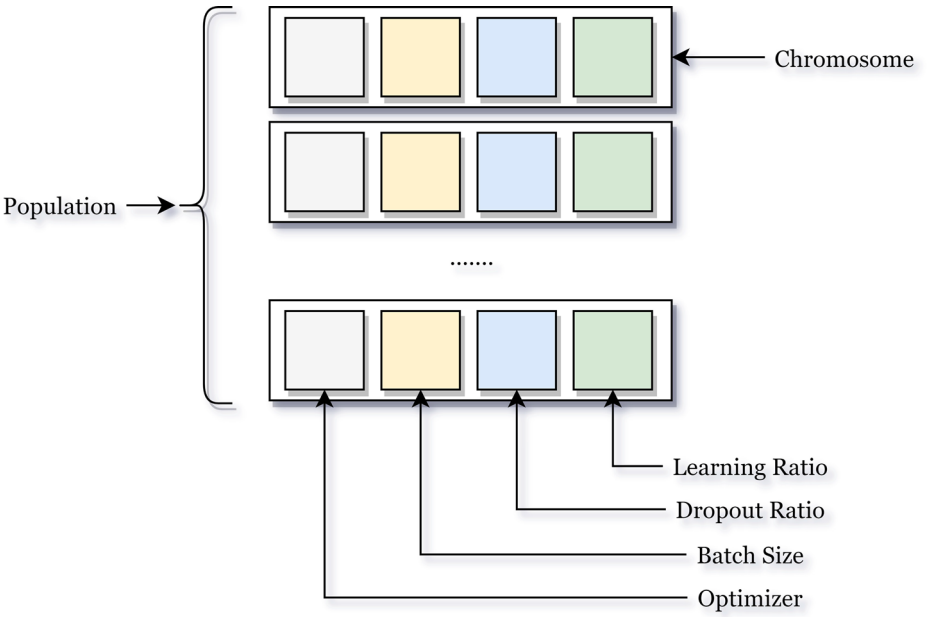

Fig. 14 Graphical representation of the GA population

highest. The used values for w_1, w_2, w_3, w_4, w_5, and w_6 in the current study are set to 0.5, 0.1, 0.1, 0.1, 0.1, and 0.1 respectively.

The values of accuracy, recall, F1-score, precision, and AUC are computed after training the transfer learning pre-trained model on the chormosome columns' values using (7), (8), (9), and (10) [109].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

where TP, TN, FP, and FN are the true positive, true negative, false positive, and false negative respectively.

Sorting After computing the objective function for each chromosome in the population set, they are sorted in a descending order concerning the objective function values.

Selection The new population is created in this phase by performing three steps. The first step is to divide the chromosomes into two halves where the first half survives and remains in the new population while the second half passes to the second step.

The second step is the crossover [98]. It is the operation of exchanging between two random parents.

To perform a crossover between two parents and generate a new chromosome, the first parent extracts its first two genes (DL weights optimizer and dropout ratio) while the second parent extracts its last two genes (batch size and TF model learning ratio) as shown in Fig. 15. The generated chromosome is passed to the third step.

The third step is the mutation [107]. It is the operation of providing diversity in the search space. It applies a small change in the mutated chromosome. It is preferred to be a low rate to avoid randomness in the search space.

The used mutation rate in the current study is set to 25% which means that only a single gene from the four genes in the mutated chromosome is changed randomly from the available options.

Algorithm 1 Proposed Approach, HMB-AHCR-DLGA, Pseudo-code.

Input: N_p, N_s // Population Size, Number of Iterations
Output: $bestCombination$
Data: Testing Dataset X
Set $O_s = [Adam, NAdam, AdaDelta, AdaGrad, AdaMax, SGD, FTRL, RMSProp]$ // DL Weights Optimizers
Set $B_s = [32, 64, 128]$ // Batch Sizes
Set $L_s = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]$ // TF Model Learning Ratios
Set $D_s = [0, 1, 2, ..., 60]$ // Dropout Ratios
Set $chromosomes = InitiatePopulation(N_p, O_s, B_s, L_s, D_s)$ // Get the Initial Population
Set $i = 1$ // Current Iteration Number
for $i \leq N_s$ **do**
    $chromosomesWithObjectiveValues = GetObjectiveValues(chromosomes)$ // Find the Chromosomes Objective Function Values of the Current Population
    $sortedChromosomes = SortChromosomes(chromosomesWithObjectiveValues)$ // Sort the Chromosomes According to the Objective Function Values
    $newPopulation = ApplySelection(sortedChromosomes, O_s, B_s, L_s, D_s)$ // Apply Crossover and Mutation and Get the New Population
    Set $chromosomes = newPopulation$ // Assign $newPopulation$ to $chromosomes$
    Set $i = i + 1$ // Increment the Iteration Number
$chromosomesWithObjectiveValues = GetObjectiveValues(chromosomes)$ // Find the Chromosomes Objective Function Values of the Last Population
$sortedChromosomes = SortChromosomes(chromosomesWithObjectiveValues)$ // Sort the Chromosomes According to the Objective Function Values
$bestChromosome = sortedChromosomes[0]$ // The First Chromosome is the Best with the Heighest Objective Function Value
Set $bestCombination = bestChromosome$

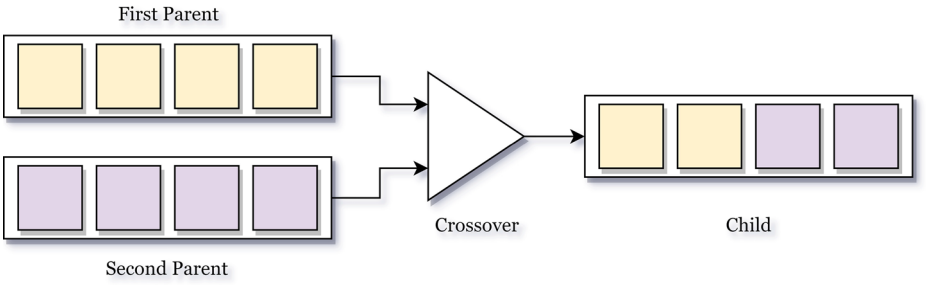

Fig. 15 Graphical representation of the GA crossover process

Iterations The objective function calculation, sorting, selection, crossover, and mutation are computed iteratively for N_s times. Algorithm 1 summarizes the proposed approach.

4 Experiments, results, and discussion

The experiments in this section are divided into two types. The first type is experiments related to the proposed native CNN models while the second type is experiments related to the proposed transfer learning and genetic algorithm, HMB-AHCR-DLGA, approach.

Table 5 summarizes the experiments' configurations. They with their corresponding results will be discussed in the following subsections.

4.1 Dataset

A large and public dataset named "HMBD" [20] is used in the current study. It is composed of 54,115 images in which each is of size 32 × 32 pixels. It is partitioned into 115 classes and used for both training and testing. It can be downloaded and used from https://github.com/hossambalaha/hmbd-v1 or https://www.kaggle.com/hossammbalaha/hmbd-v1-arabic-handwritten-characters-dataset

Data augmentation is applied to the dataset to increase the accuracy and reduce the overfitting. The data augmentation used methods are image rotation, shifting, zooming, and shearing [97].

4.2 CNN architectures

The objective is to choose the best model that fits the dataset well. As mentioned, many try-and-error trials in the network configuration tuning mechanism are performed. This resulted in 14 different CNN architectures, CNN-1 to CNN-14, which are compiled and trained after that.

The main difference between them is the hierarchy of building the architecture layers. The best CNN architecture is selected from them. Then, different combinations of weight initializers and optimizers are applied to the best-selected architectures. The later process aims to report the most suitable weight initializers and optimizers. The training process is performed on Google Colab [24] with its Graphical Processing Unit (GPU) environment.

Table 5 Experiments configurations summarization

Approach	Configuration	Values
CNN Architectures	Dataset	HMBD [20]
	Image Size	(32, 32) (Grayscale)
	Models	CNN-1 to CNN-14 (14 Models)
	DL Weight Initializers	8 Initializers: Glorot Uniform [96], Random Normal, Random Uniform, Truncated Normal, Variance Scaling [47], Orthogonal LeCun Uniform, Glorot Normal [120], and He Uniform
	Weight Optimizers	7 Optimizers: Adam, NAdam [119], AdaGrad [88, 122], AdaDelta [133], AdaMax [60], RMSProp, and SGD
	Number of Epochs	50
	Batch Size	32
	Hidden Activation Function	ReLU
	Output Activation Function	SoftMax
	Training Environment	Google Colab [24]
HMB-AHCR-DLGA	Dataset	HMBD [20]
	Image Size	(32, 32, 3) (RGB)
	Models	VGG16 [113], VGG19 [113], and MobileNetV2 [104]
	Pre-trained DL Weights	ImageNet [33]
	DL Weight Optimizers (O_s)	8 Optimizers: Adam, NAdam [119], AdaGrad [88, 122], AdaDelta [133], AdaMax [60], RMSProp, SGD, and FTRL [82]
	Number of Epochs	64
	Batch Sizes (B_s)	[32, 64, 128]
	TF Model Learn Ratios (L_s)	[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
	Dropout Ratios (D_s)	[1, 2, ..., 60]
	Number of GA Iterations (N_s)	10
	Training Environments	Google Colab [24] and Toshiba Qosmio X70-A

To select the best and leading architectures, the 14 suggested architectures, CNN-1 to CNN-14, are trained and the best three are selected using two scenarios (1) without data augmentation and (2) with data augmentation.

The number of epochs and batch size are 50 and 32 respectively. The default optimizer and weight initializer are Adam and Glorot Uniform [96] respectively.

Tables 6 and 7 show the reported results for the 14 different CNN architectures without data augmentation and with data augmentation respectively.

The CNN architectures CNN-12, CNN-14, and CNN-11 reported the top-3 testing accuracy architectures for the first scenario (without data augmentation), and CNN-12, CNN-11, and CNN-5 reported the top-3 testing accuracy architectures for the second scenario (with data augmentation).

Architecture CNN-9 failed to reach valuable accuracy after applying data augmentation on it. This abnormal result means that the model fell in the local minima and could not get out from it.

Table 6 Results of the 14 CNN architectures after training on HMBD without data augmentation

Architecture no.	Training accuracy	Training loss	Testing accuracy	Testing loss
1	84.92%	0.46	83.04%	0.59
2	84.99%	0.44	81.67%	0.66
3	89.23%	0.31	76.98%	0.97
4	77.87%	0.68	76.20%	0.80
5	87.34%	0.39	85.14%	0.50
6	79.23%	0.70	77.35%	0.95
7	84.86%	0.47	80.49%	0.67
8	75.79%	0.76	62.45%	1.50
9	71.89%	0.85	69.48%	1.14
10	96.34%	0.11	74.28%	0.99
11	96.61%	0.10	85.22%	0.57
12	**88.37%**	**0.34**	**88.06%**	**0.39**
13	83.48%	0.51	80.45%	0.70
14	86.81%	0.40	86.29%	0.49

The bold entries reflect the best reported results concerning the testing accuracies

Figure 16 shows the curve of the 14 architectures in the two scenarios. The blue curve is for the first scenario while the orange curve is for the second scenario. The x-axis represents the 14 architectures and the y-axis represents the testing accuracies.

The top-3 CNN architectures in the second scenario are trained after that on sets of different optimizers and weight initalizers to report the best combinations. The used DL

Table 7 Results of the 14 CNN architectures after training on HMBD with data augmentation

Architecture no.	Training accuracy	Training loss	Testing accuracy	Testing loss
1	66.15%	1.16	78.12%	0.70
2	77.27%	0.74	84.05%	0.53
3	77.87%	0.70	79.74%	0.73
4	63.74%	1.26	72.07%	0.95
5	74.03%	0.90	85.01%	0.50
6	59.78%	1.52	75.71%	0.81
7	68.90%	1.08	80.87%	0.64
8	66.77%	1.10	66.87%	1.19
9	1.70%	4.74	1.69%	4.74
10	90.01%	0.29	79.46%	0.92
11	92.07%	0.23	87.31%	0.45
12	**82.81%**	**0.56**	**88.22%**	**0.37**
13	75.12%	0.85	83.15%	0.55
14	75.78%	0.78	76.22%	0.82

The bold entries reflect the best reported results concerning the testing accuracies

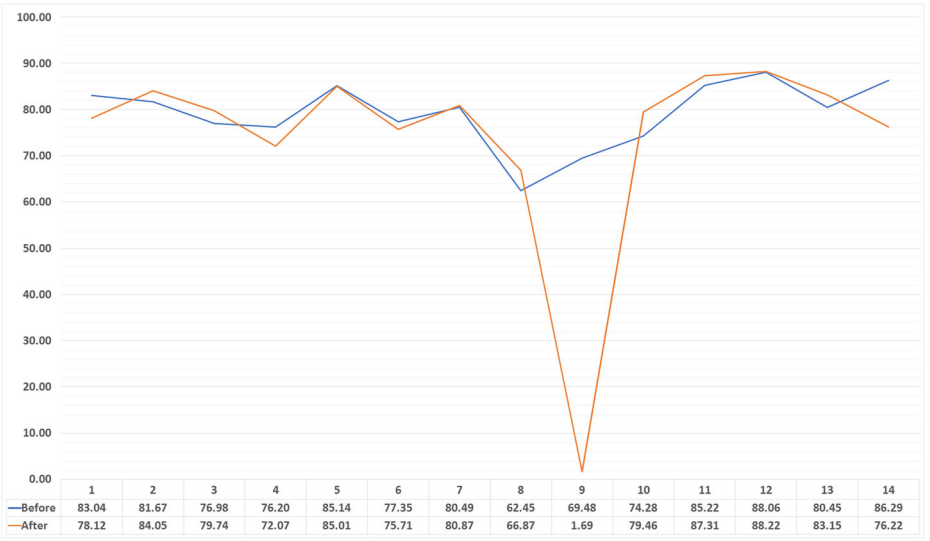

Fig. 16 Testing accuracies before and after data augmentation of the 14 architectures

optimizers are Adam, NAdam [119], AdaGrad [88, 122], AdaDelta [133], AdaMax [60], RMSProp, and SGD.

The used weight initalizers are: Glorot Uniform [96], Random Normal, Random Uniform, Truncated Normal, Variance Scaling [47], Orthogonal LeCun Uniform, Glorot Normal [120], and He Uniform.

Tables 8, 9, and 10 report the top-10 training experiments (concerning the testing accuracies) performed on the CNN architectures CNN-12, CNN-11, and CNN-5 respectively using different weight initializers and optimizers applied on the augmented version of the HMBD dataset. The tables are ordered in descending order according to the testing accuracies.

The CNN architecture, CNN-12, reported the highest testing accuracy value of 88.17% with a testing loss value of 0.38 using AdaMAx optimizer and Glorot Normal weight initializer. Figure 17 shows a graphical representation of the testing and training accuracies

Table 8 Training the CNN architecture no. 12 using different optimizers and weights initializers on HMBD with data augmentation

Optimizer	Weight initializer	Training accuracy	Training loss	Testing accuracy	Testing loss
AdaMax	Glorot Normal	82.33%	0.58	88.17%	0.38
AdaMax	Variance Scaling	82.27%	0.58	88.08%	0.38
Adam	Glorot Uniform	82.76%	0.56	87.98%	0.38
NAdam	Glorot Uniform	82.48%	0.57	87.95%	0.38
AdaMax	He Normal	81.94%	0.59	87.90%	0.38
AdaMax	Glorot Uniform	82.25%	0.58	87.90%	0.38
AdaMax	LeCun Uniform	82.19%	0.58	87.89%	0.38
AdaMax	Random Uniform	82.21%	0.58	87.87%	0.39
AdaMax	Truncated Normal	82.24%	0.58	87.80%	0.38
AdaMax	Orthogonal	82.05%	0.59	87.77%	0.38

Table 9 Training the CNN architecture No. 11 using different optimizers and weights initializers on HMBD with data augmentation

Optimizer	Weight initializer	Training accuracy	Training loss	Testing accuracy	Testing loss
Adam	He Uniform	92.30%	0.22	87.70%	0.46
AdaMax	Truncated Normal	91.70%	0.24	87.52%	0.44
Adam	Truncated Normal	92.44%	0.22	87.51%	0.43
Adam	He Normal	92.26%	0.22	87.44%	0.44
Adam	LeCun Uniform	92.38%	0.22	87.44%	0.42
Adam	Random Normal	92.28%	0.22	87.38%	0.45
Adam	LeCun Normal	92.51%	0.22	87.21%	0.45
AdaMax	Random Normal	91.56%	0.25	87.19%	0.44
Adam	Variance Scaling	92.37%	0.22	87.16%	0.42
SGD	Random Uniform	90.61%	0.27	87.15%	0.44

for CNN-12. The graph shows that the testing accuracy precedes the training accuracy and hence there is an underfitting with a maximum difference of 5.96%.

The CNN architecture, CNN-11, reported the highest testing accuracy value of 87.70% with a testing loss value of 0.46 using Adam optimizer and He Uniform weight initializer. Figure 18 shows a graphical representation of the testing and training accuracies for CNN-11. The graph shows that the training accuracy precedes the testing accuracy and hence there is an overfitting with a maximum difference of 5.30%.

The CNN architecture, CNN-5, reported the highest testing accuracy value of 91.96% with a testing loss value of 0.23 using SGD optimizer and LeCun Uniform weight initializer. Figure 19 shows a graphical representation of the testing and training accuracies for CNN-5. The graph shows that the testing accuracy precedes the training accuracy and hence there is an underfitting with a maximum difference of 1.94%.

This reports the CNN architecture, CNN-5, as the best architecture among others as it achieved the highest testing accuracy of 91.96%.

Table 10 Training the CNN architecture no. 5 using different optimizers and weights initializers on HMBD with data augmentation

Optimizer	Weight Initializer	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
SGD	LeCun Uniform	90.62%	0.26	91.96%	0.23
AdaMax	Truncated Normal	90.49%	0.30	91.89%	0.25
AdaMax	Glorot Uniform	90.49%	0.29	91.86%	0.24
AdaMax	LeCun Uniform	90.10%	0.31	91.80%	0.25
AdaMax	Random Uniform	89.88%	0.32	91.67%	0.25
AdaMax	Variance Scaling	90.51%	0.29	91.65%	0.24
AdaMax	Random Normal	90.10%	0.31	91.62%	0.25
AdaMax	LeCun Normal	89.97%	0.31	91.57%	0.25
AdaMax	Glorot Normal	90.20%	0.30	91.47%	0.26
AdaMax	He Normal	89.47%	0.33	91.41%	0.26

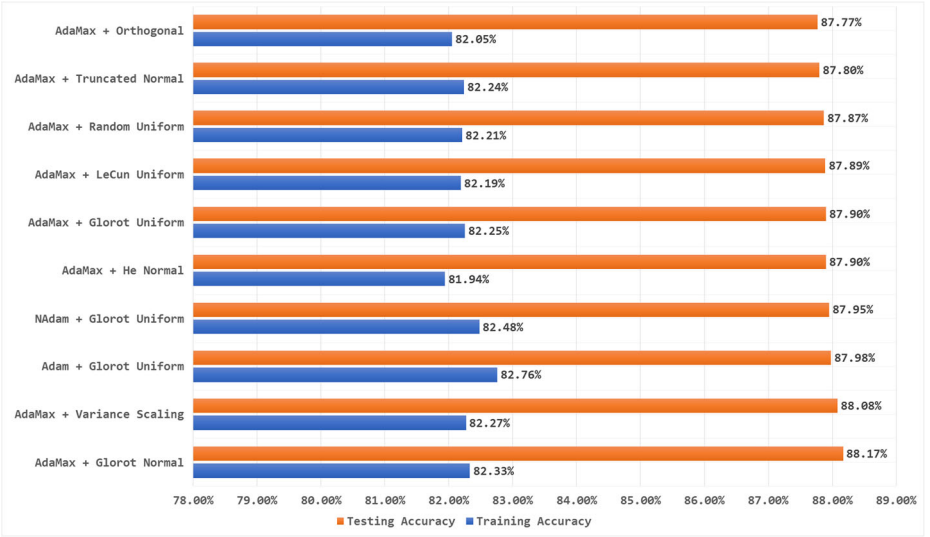

Fig. 17 Graphical representation of the testing and training accuracies for CNN-12

4.3 The HMB-AHCR-DLGA

The objective is to optimize the hyperparameters to evaluate the best combination set. VGG16 [113], VGG19 [113], and MobileNetV2 [104] pre-trained CNN models with the ImageNet [33] weights. The used DL optimizers are Adam, NAdam [119], AdaGrad [88, 122], AdaDelta [133], AdaMax [60], RMSProp, SGD, and FTRL [82].

The images are used in the RGB format with the size of $32 \times 32 \times 3$. The number of epochs and number of GA iterations are set to 64 and 10 respectively. The training process is performed on Google Colab [24] with its GPU environment and

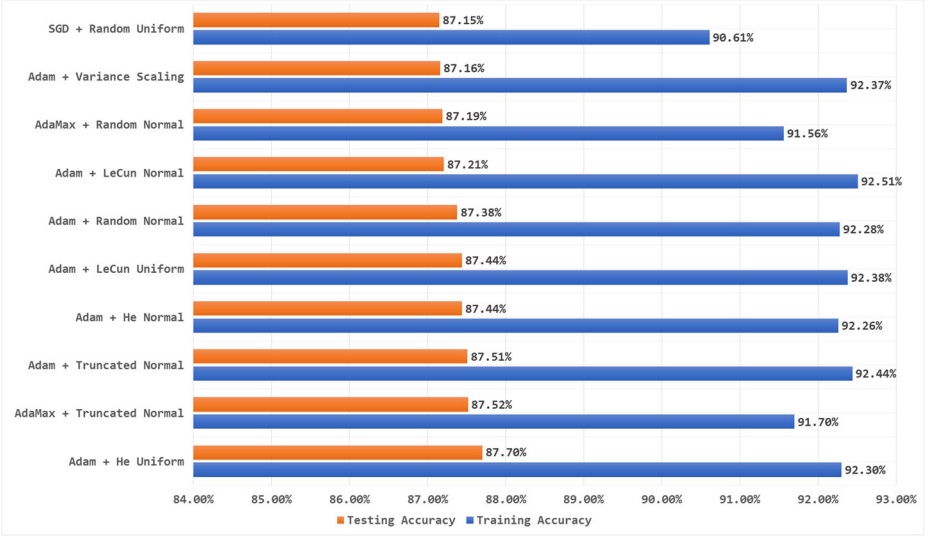

Fig. 18 Graphical representation of the testing and training accuracies for CNN-11

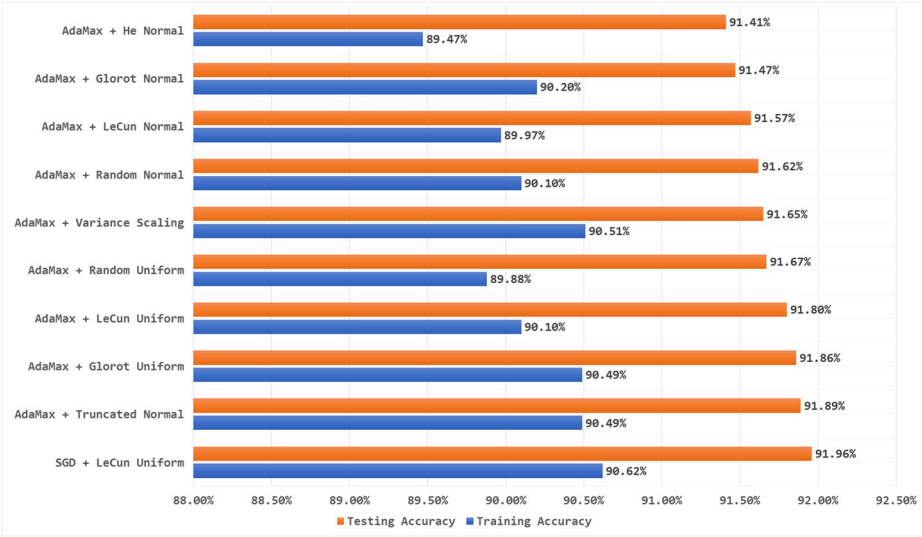

Fig. 19 Graphical representation of the testing and training accuracies for CNN-5

Toshiba Qosmio X70-A with 32 GB of RAM and NVIDIA GeForce GTX 770M GPU. The used batch sizes are [32, 64, 128]. The used TF model learn ratios are [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]. The used dropout ratios are [1, 2, ..., 60].

Five experiments are performed as follows three experiments are performed on VGG16, one on VGG19, and one on MobileNetV2. Table 11 reports the best combinations with the corresponding hyperparameters for each experiment.

AdaMax is reported to be the best DL weight optimizer in three experiments. The batch size of 64 is reported to be the best in three experiments. The dropout ratio of 10% is reported to be the best in three experiments.

The TF model learning ratio 60% is reported to be the best in three experiments. Two experiments using VGG16 (first and third) report testing accuracies better than the highest testing accuracy in the native CNN models experiments (91.96%).

Table 11 Experiments results using the proposed HMB-AHCR-DLGA approach

Pre-trained TF Model	First VGG16	Second VGG16	Third VGG16	VGG19	MobileNetV2
DL Weights Optimizer	AdaMax	AdaGrad	AdaMax	SGD	AdaMax
Batch Size	128	64	64	32	64
Dropout Ratio	10%	10%	0%	30%	10%
TF Model Learn Ratio	60%	60%	60%	55%	50%
Objective Function Value	84.45%	81.73%	84.52%	82.87%	73.23%
Testing Loss	0.374	0.366	0.393	0.424	0.725
Testing Accuracy	92.85%	89.28%	92.88%	90.91%	78.91%
Testing F1-Score	92.97%	89.59%	93.21%	91.07%	79.13%
Testing Precision	93.41%	91.29%	93.71%	91.86%	83.24%
Testing Recall	92.54%	88.00%	92.74%	90.36%	75.51%
Testing AUC	98.71%	99.24%	98.57%	98.51%	98.49%

The lowest testing loss is 0.374 by the first VGG16 experiment. The highest testing accuracy is 92.88% by the third VGG16 experiment. The highest testing F1-score is 93.21% by the third VGG16 experiment. The highest testing precision is 93.71% by the third VGG16 experiment.

The highest testing recall is 92.74% by the third VGG16 experiment. The highest testing AUC is 99.24% by the second VGG16 experiment.

As mentioned, the highest testing accuracy is 92.88% which is achieved by the third VGG16 experiment using AdaMax weights optimizer, 64 batch size value, 0% dropout ratio, and 60% TF model learning ratio. It can be considered as the best combination using the VGG16 pre-trained TF model for the HMBD dataset.

It worth mentioning that, the HMB-AHCR-DLGA approach reported higher testing accuracy by the third VGG16 (92.74%) than the suggested abstract CNN model, CNN-5 (91.96%). However, the CNN-5 architecture is less complex than the third VGG16 concerning the number of parameters and neurons. Hence, it requires less memory, processing, and time in the prediction phase.

As the difference between them is less than 1%, we can select the third VGG16 if we are targeting the prediction accuracy and we can select the CNN-5 architecture if we are targeting the prediction speed.

5 Conclusions and future work

Handwritten Character Recognition for Arabic characters is an active research area that always needs an improvement in results. The study presented different AHCR systems concerning the related work. In this study, CNNs are used for the recognition of Arabic handwritten characters problem using the 28 Arabic characters. CNN is a well-known state-of-the-art algorithm for classifying images and big data.

The deep learning-based AHCR system is developed. The study presented the different challenges related to text segmentation by describing a set of solutions that can be further improved in future studies.

The segmented elements should be entered into the recognition phase to be converted from handwritten characters format into the printed ones. The system can be divided into two main phases. The first one is to divide the given handwritten text into segments. The second is to convert the segments into printed ones.

From the trials, it is very difficult to use a general formula to produce text segments due to the cursive nature and ligatures of the Arabic language. Handwriting varies from a person to another, different lighting, and many other presented reasons. The solution for each of these challenges has been suggested.

In the second phase, the classification phase, different experiments in two paths were performed and the results are reported.

For the first experimental path, 14 CNN architectures were suggested after lots of trials. The training was performed on the HMBD dataset. Two scenarios (1) without data augmentation and (2) with data augmentation were applied to the architectures.

The testing accuracy was used as the performance metric to judge the best architecture. Different optimization methods were also applied to increase the CNN performance. The top-3 in each scenario were reported. The CNN architectures CNN-12, CNN-14, and CNN-11 reported the top-3 testing accuracies for the first scenario, and CNN-12, CNN-11, and CNN-5 reported the top-3 testing accuracies for the second scenario.

The top-3 CNN architectures were trained after that on sets of different optimizers and weight initializers. The CNN architecture, CNN-12, reported the 88.17% as the highest

testing accuracy value with a testing loss value of 0.38 using AdaMAx optimizer and Glorot Normal weight initializer. The CNN architecture, CNN-11, reported 87.70% as the highest testing accuracy value with a testing loss value of 0.46 using Adam optimizer and He Uniform weight initializer.

The CNN architecture, CNN-5, reported 91.96% as the highest testing accuracy value with a testing loss value of 0.23 using SGD optimizer and LeCun Uniform weight initializer. The CNN architecture, CNN-5, reported the best architecture among others as it achieved the highest testing accuracy value 91.96%

For the second experimental path, a TF and GA approach (named "HMB-AHCR-DLGA") was proposed to optimize the training parameters and hyperparameters. The HMBD dataset with data augmentation was used in the training. Accuracy, recall, F1-score, AUC, and precision were used as the performance metrics to judge the best pre-trained CNN architecture.

The highest testing accuracy using the HMB-AHCR-DLGA approach was 92.88% which was achieved by the third VGG16 experiment using AdaMax weights optimizer, 64 batch size value, 0% dropout ratio, and 60% TF model learning ratio. It could be considered as the best combination using the VGG16 pre-trained CNN model.

Even though there is a lot of work for the recognition of handwritten characters for English, Chinese, French, and some other Indian languages, only a little work was done for the Arabic language. Due to the lack of research work in the area, there was a big challenge to get a dataset for the Arabic language.

In future studies, the plan is to (1) work on improving the performance of the handwritten text segmentation phase, (2) work on Arabic handwritten word recognition using deep learning architectures, (3) apply the system on different available datasets, and (4) try different machines learning or deep learning approaches.

Appendix A

A. 1 Table of abbreviations

Table 12 presents the abbreviations.

Table 12 Table of abbreviations

Abbreviation	Discussion
Adam	Adaptive Moment Optimization Algorithm
AHCR	Arabic Handwritten Character Recognition
AUC	Area Under Curve
CRR	Correct Classification Rate
CNN	Convolutional Neural Network
DL	Deep Learning
ECR	Error Classification Rate
GA	Genetic Algorithm
GPU	Graphical Processing Unit
NP-hard	Non-deterministic Polynomial-time Hard
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
RMSS	Root Mean Square Summation
TF	Tranfer Learning
WSA	Weighted Sum Approach

Author agreement statement We the undersigned declare that this manuscript is original, has not been published before, and is not currently being considered for publication elsewhere. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. We understand that the "Corresponding Author" is the sole contact for the editorial process. He is responsible for communicating with the other authors about progress, submissions of revisions, and final approval of proofs.

Funding No funding was received for this work.

Declarations

Conflict of Interests No conflict of interest exists. We wish to confirm that, there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Intellectual Property We confirm that, we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

Authorship We confirm that the manuscript has been read and approved by all named authors. We confirm that the order of authors listed in the manuscript has been approved by all named authors.

Contact with the Editorial Office The "Corresponding Author" is declared on the title page. This author submitted this manuscript using his account in the editorial submission system. (A) We understand that the "Corresponding Author" is the sole contact for the editorial process (including the editorial submission system and direct communications with the office). He is responsible for communicating with the other authors about progress, submissions of revisions, and final approval of proofs. (B) We confirm that the email address shown below is accessible by the "Corresponding Author", is the address to which "Corresponding Author"'s editorial submission system account is linked, and has been configured to accept email from the editorial office (Email: **hossam.m.balaha@mans.edu.eg**).

References

1. Abuhaiba IS, Holt MJ, Datta S (1998) Recognition of off-line cursive handwriting. Comput Vis Image Underst 71(1):19–38
2. Acharya S, Pant AK, Gyawali PK (2015) Deep learning based large scale handwritten devanagari character recognition. In: 2015 9th International conference on software, knowledge, information management and applications (SKIMA). IEEE, pp 1–6
3. Ahmed R et al (2020) Offline arabic handwriting recognition using deep machine learning: A review of recent advances. In: Ren J et al (eds) Advances in brain inspired cognitive systems. Springer International Publishing, Cham, pp 457-468
4. Akhand M, Ahmed M, Rahman MH (2016) Convolutional neural network based handwritten bengali and bengali-english mixed numeral recognition. Int J image Graph Signal Process 8(9):40
5. Al-Helali BM, Mahmoud SA (2017) Arabic online handwriting recognition (aohr): A survey. ACM Comput Surv 50(3)
6. Al-Shaher AA, Hancock ER (2003) Learning mixtures of point distribution models with the em algorithm. Pattern Recogn 36(12):2805–2818
7. Al-Taani AT, Al-Haj S (2010) Recognition of on-line arabic handwritten characters using structural features. J Pattern Recognit Res 5(1):23–37
8. AlKhateeb JH, Ren J, Jiang J, Al-Muhtaseb H (2011) Offline handwritten arabic cursive text recognition using hidden markov models and re-ranking. Pattern Recogn Lett 32(8):1081–1088

9. Ali AAA, M Suresha (2019) Arabic handwritten character recognition using machine learning approaches. In: 2019 Fifth international conference on image information processing (ICIIP). pp 187–192
10. Ali A, Zhu Y, Chen Q, Yu J, Cai H (2019) Leveraging spatio-temporal patterns for predicting citywide traffic crowd flows using deep hybrid neural networks. In: 2019 IEEE 25th international conference on parallel and distributed systems (ICPADS). IEEE, pp 125–132
11. Ali A, Zhu Y, Zakarya M (2021) A data aggregation based approach to exploit dynamic spatio-temporal correlations for citywide crowd flows prediction in fog computing. Multimed Tools Appl :1–33
12. Althobaiti H, Chao Lu (2017) A survey on arabic optical character recognition and an isolated handwritten arabic character recognition algorithm using encoded freeman chain code. In: 2017 51st Annual conference on information sciences and systems (CISS). pp 1–6
13. Altwaijry N, Al-Turaiki I (2020) Arabic handwriting recognition system using convolutional neural network. Neural Comput Appl
14. Amin A (1997) Arabic character recognition. In: Handbook of character recognition and document image analysis. World Scientific. pp 397–420
15. Amin A (2003) Recognition of hand-printed characters based on structural description and inductive logic programming. Pattern Recognit Lett 24(16):3187–3196
16. Athoillah M, Putri RK (2019) Handwritten arabic numeral character recognition using multi kernel support vector machine. KINETIK: Game technology, information system, computer network, computing, electronics, and control 4(2):99–106
17. Bafjaish SS, Azmi MS, Al-Mhiqani MN, Sheikh AA (2020) Skew correction for mushaf al-quran: a review. Indones J Electr Eng Comput Sci 17(1):516–523
18. Bai J, Chen Z, Feng B, Xu B (2014) Image character recognition using deep convolutional neural network learned from different languages. In: 2014 IEEE International conference on image processing (ICIP). IEEE, pp 2560–2564
19. Balaha HM, Ali HA, Badawy M (2020) Automatic recognition of handwritten arabic characters: a comprehensive review. Neural Comput Appl :1–24
20. Balaha HM, Ali HA, Saraya M, Badawy M (2020) A new arabic handwritten character recognition deep learning system (ahcr-dls). Neural Comput Appl :1–43
21. Baldi P (2011) Autoencoders, unsupervised learning and deep architectures. In: Proceedings of the 2011 international conference on unsupervised and transfer learning workshop - Volume 27, UTLW'11. (JMLR.org), pp 37–50
22. Bengio Y, Ca M (2015) Rmsprop and equilibrated adaptive learning rates for nonconvex optimization. arXiv:1502.04390
23. Bernardin K, Stiefelhagen R (2008) Evaluating multiple object tracking performance: the clear mot metrics. EURASIP J Image Vid Process 2008:1–10
24. Bisong E (2019) Google colaboratory. In: Building machine learning and deep learning models on google cloud platform. Springer, pp 59–64
25. Bradski G (2008) Kaehler A. O'Reilly Media, Inc., Newton
26. Changming S, Deyi S (1997) Skew and slant correction for document images using gradient direction. In: Proceedings of the fourth international conference on document analysis and recognition. vol 1, pp 142–146
27. Chollet F (2016) Xception: Deep learning with depthwise separable convolutions. arXiv:1610.02357
28. Christlein V et al (2019) Deep generalized max pooling. In: 2019 International conference on document analysis and recognition (ICDAR). pp 1090–1096
29. Chua LO, Roska T (1993) The cnn paradigm. IEEE Trans Circ Syst I Fund Theory Appl 40(3):147–156
30. Ciregan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In: 2012 IEEE conference on computer vision and pattern recognition. IEEE, pp 3642–3649
31. Cogswell M, Ahmed F, Girshick R, Zitnick L, Batra D (2015) Reducing overfitting in deep networks by decorrelating representations. arXiv:1511.06068
32. Dai Y et al (2018) Fused text segmentation networks for multi-oriented scene text detection. In: 2018 24th international conference on pattern recognition (ICPR). pp 3604–3609
33. Deng J et al (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on computer vision and pattern recognition. pp 248–255
34. Duan K, Keerthi SS, Chu W, Shevade SK, Poo AN (2003) Multi-category classification by soft-max combination of binary classifiers. In: International workshop on multiple classifier systems. Springe), pp 125–134
35. El-Desouky A, Salem M, El-Gwad AA, Arafat H (1991) A handwritten arabic character recognition technique for machine reader. In: Third international conference on software engineering for real time systems, (IET). pp 212–216

36. El-Sawy A, EL-Bakry H, Loey M (2017) Cnn for handwritten arabic digits recognition based on lenet-5. In: Hassanien AE, Shaalan K, Gaber T, Azar AT, Tolba MF (eds) Proceedings of the international conference on advanced intelligent systems and informatics 2016. Springer International Publishing, Cham, pp 566–575
37. ElAdel A, Ejbali R, Zaied M, Amar CB (2015) Dyadic multi-resolution analysis-based deep learning for arabic handwritten character classification. In: 2015 IEEE 27th International conference on tools with artificial intelligence (ICTAI). IEEE, pp 807–812
38. Elarian Y, Ahmad I, Awaida S, Al-Khatib W, Zidouri A (2015) Arabic ligatures: analysis and application in text recognition. In: 2015 13th International conference on document analysis and recognition (ICDAR). IEEE, pp 896–900
39. Elarian Y, Ahmad I, Zidouri A, Al-Khatib WG (2019) Lucidah ligative and unligative characters in a dataset for arabic handwriting. Int J Adv Comput Sci Appl 10(8)
40. Elleuch M, Tagougui N, Kherallah M (2015) Arabic handwritten characters recognition using deep belief neural networks. In: 2015 IEEE 12th International multi-conference on systems, signals devices (SSD15). pp 1–5
41. Elzobi M, Al-Hamadi A, Al Aghbari Z, Dings L (2013) Iesk-ardb: a database for handwritten arabic and an optimized topological segmentation approach. Int J Doc Anal Recognit (IJDAR) 16(3):295–308
42. Farooq F, Venu Govindaraju, Perrone M (2005) Pre-processing methods for handwritten arabic documents. In: Eighth international conference on document analysis and recognition (ICDAR'05), vol 1. pp 267–271
43. Firdaus FI, Khumaini A, Utaminingrum F (2017) Arabic letter segmentation using modified connected component labeling. In: 2017 International conference on sustainable information engineering and technology (SIET). pp 392–397
44. Gardiner AH (1916) The egyptian origin of the semitic alphabet. J Egypt Archaeol 3(1):1–16
45. Genlin J (2004) Survey on genetic algorithm [j]. Comput Appl Softw 2(1):69–73
46. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
47. Grahl J, Bosman PA, Rothlauf F (2006) The correlation-triggered adaptive variance scaling idea. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. pp 397–404
48. Ham FM, Han S (1996) Classification of cardiac arrhythmias using fuzzy artmap. IEEE Trans Biomed Eng 43(4):425–429
49. Hämäläinen W (2006) Class np, np-complete, and np-hard problems
50. Hara K, Saito D, Shouno H (2015) Analysis of function of rectified linear unit used in deep learning. In: 2015 international joint conference on neural networks (IJCNN). IEEE, pp 1–8
51. He M, Zhang S, Mao H, Jin L (2015) Recognition confidence analysis of handwritten chinese character with cnn. In: 2015 13th International conference on document analysis and recognition (ICDAR). IEEE, pp 61–65
52. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv:1512.03385
53. Hesterman JY, Caucci L, Kupinski MA, Barrett HH, Furenlid LR (2010) Maximum-likelihood estimation with a contracting-grid search algorithm. IEEE Trans Nuclear Sci 57(3):1077–1084
54. Heutte L, Moreau JV, Paquet T, Lecourtier Y, Olivier C (1996) Combining structural and statistical features for the recognition of handwritten characters. In: Proceedings of 13th International conference on pattern recognition. IEEE, vol 2, pp 210-214
55. Hifny Y (2019) Open vocabulary arabic diacritics restoration. IEEE Signal Process Lett 26(10):1421–1425
56. Hinton GE (2009) Deep belief networks. Scholarpedia 4(5):5947
57. Hirose Y, Yamashita K, Hijiya S (1991) Back-propagation algorithm which varies the number of hidden units. Neural Netw 4(1):61–66
58. Howard AG et al (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
59. Huang G, Liu Z, Weinberger KQ (2016) Densely connected convolutional networks. arXiv:1608.06993
60. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y (2016) Binarized neural networks. In: Advances in neural information processing systems. pp 4107–4115
61. Jain AK, Duin RPW, Mao J (2000) Statistical pattern recognition: A review. IEEE Trans Pattern Anal Mach Intell 22(1):4–37
62. Javed F (2013) Arabic and english phonetics: A comparative study. Criterion 4(4):1–13
63. Jiang Y et al (2018) Expert feature-engineering vs. deep neural networks: which is better for sensor-free affect detection? In: International conference on artificial intelligence in education. Springer, pp 198–211

64. Junker M, Hoch R, Dengel A (1999) On the evaluation of document analysis components by recall, precision, and accuracy. In: Proceedings of the fifth international conference on document analysis and recognition. ICDAR'99 (Cat. No. PR00318). IEEE, pp 713–716
65. Kar R et al (2019) Novel approaches towards slope and slant correction for tri-script handwritten word images. Imaging Sci J 67(3):159–170
66. Karim A, Mahdi B, Abdullah A (2019) Writer identification based on arabic handwriting recognition by using speed up robust feature and k- nearest neighbor classification. J Univ Babylon Pure Appl Sci 27:1–10
67. Kaye AS (2003) Arabic. In: The major languages of South Asia, the Middle East and Africa. (Routledge), pp 144–161
68. Kef M, Chergui L, Chikhi S (2016) A novel fuzzy approach for handwritten arabic character recognition. Pattern Anal Appl 19(4):1041–1056
69. Khuman YLK, Devi HM, Singh NA (2021) Entropy-based skew detection and correction for printed meitei/meetei script ocr system. Mater Toda Proc 37:2666–2669
70. Kim Y (2014) Convolutional neural networks for sentence classification. arXiv:1408.5882
71. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980
72. Kumar T, Verma K (2010) A theory based on conversion of rgb image to gray image. Int J Comput Appl 7(2):7–10
73. Lamtougui H, Moubtahij HE, Fouadi H, Yahyaouy A, Satori K (2020) Offline arabic handwriting recognition using deep learning: Comparative study. In: 2020 International conference on intelligent systems and computer vision (ISCV). pp 1–8
74. Lawgali A, Angelova M, Bouridane A (2013) Hacdb: Handwritten arabic characters database for automatic character recognition. In: European workshop on visual information processing (EUVIP). IEEE, pp 255–259
75. Lawgali A, Angelova M, Bouridane A (2014) A framework for arabic handwritten recognition based on segmentation. Int J Hybrid Inf Technol 7:413–428
76. LeCun Y, Bengio Y et al (1995) Convolutional networks for images, speech, and time series. In: The handbook of brain theory and neural networks, vol 3361, p 1995
77. Liang Y, Wang J, Zhou S, Gong Y, Zheng N (2016) Incorporating image priors with deep convolutional neural networks for image super-resolution. Neurocomputing 194:340–347
78. Liu B et al (2017) Supervised deep feature extraction for hyperspectral image classification. IEEE Trans Geosci Remote Sens 56(4):1909–1921
79. Lu J et al (2015) Transfer learning using computational intelligence: A survey, vol 80, pp 14–23. 25th anniversary of Knowledge-Based Systems
80. Maitra DS, Bhattacharya U, Parui SK (2015) Cnn based common approach to handwritten character recognition of multiple scripts. In: 2015 13th International conference on document analysis and recognition (ICDAR). IEEE, pp 1021–1025
81. Marler RT, Arora JS (2010) The weighted sum method for multi-objective optimization: new insights. Struct Multidiscipl Optim 41(6):853–862
82. McMahan HB et al (2013) Ad click prediction: a view from the trenches. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp 1222–1230
83. Miikkulainen R, et al. (2019) Evolving deep neural networks. In: Artificial intelligence in the age of neural networks and brain computing. Elsevier, pp 293–312
84. Mirjalili S (2019) Genetic algorithm. In: Evolutionary algorithms and neural networks. (Springer), pp 43–55
85. Motawa D, Amin A, Sabourin R (1997) Segmentation of arabic cursive script. In: Proceedings of the fourth international conference on document analysis and recognition, vol 2. pp 625–628
86. Motwani MC, Gadiya MC, Motwani RC, Harris FC Jr (2004) Survey of image denoising techniques
87. Mozaffari S, Faez K, Ziaratban M (2005) Character representation and recognition using quad tree-based fractal encoding scheme. In: Eighth international conference on document analysis and recognition (ICDAR'05). vol 2, pp 819–823
88. Mukkamala MC, Hein M (2017) Variants of rmsprop and adagrad with logarithmic regret bounds. arXiv:1706.05507
89. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp 807–814
90. Naz S (2015) Segmentation techniques for recognition of arabic-like scripts: A comprehensive survey. Springer J Educ Inf Technol :21
91. Nwankpa C, Ijomah W, Gachagan A, Marshall S (2018) Activation functions: Comparison of trends in practice and research for deep learning. arXiv:1811.03378

92. Pak I, Teh PL (2018). In: Zelinka I VasantP Duy VH (ed) Text segmentation techniques: a critical review. Springer International Publishing, Cham, pp 167–181
93. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
94. Parvez MT, Mahmoud SA (2013) Offline arabic handwritten text recognition: a survey. ACM Comput Surv (CSUR) 45(2):1–35
95. Pechwitz M et al (2002) Ifn/enit-database of handwritten arabic words. In: Proceedings of CIFED. vol 2, Citeseer, pp 127–136
96. Pedamonti D (2018) Comparison of non-linear activation functions for deep neural networks on mnist classification task. arXiv:1804.02763
97. Perez L, Wang J (2017) The effectiveness of data augmentation in image classification using deep learning. arXiv:1712.04621
98. Poon PW, Carter JN (1995) Genetic algorithm crossover operators for ordering applications. Comput Oper Res 22(1):135–147
99. Pratihar DK (2013) Soft computing: fundamentals and applications. Alpha Science International Ltd, Oxford
100. Retsö J (2013) What is arabic. In: The Oxford handbook of Arabic linguistics, vol 422, p 450
101. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv:1609.04747
102. Russakovsky O et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
103. Sahlol A, Suen C (2014) A novel method for the recognition of isolated handwritten arabic characters
104. Sandler M, Howard AG, Zhu M, Zhmoginov A, Chen L (2018) Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. arXiv:1801.04381
105. Saravanan C (2010) Color image to grayscale image conversion. In: 2010 second international conference on computer engineering and applications. IEEE, vol 2, pp 196-199
106. Scherer D, Müller A, Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. In: International conference on artificial neural networks. Springer, pp 92–101
107. Schmitt LM (2001) Theory of genetic algorithms. Theor Comput Sci 259(1-2):1–61
108. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
109. Set T (2021) Precision recall f1-score auc accuracy, vol 16, p 12
110. Shams M, Elsonbaty AZW (2020) Arabic handwritten character recognition based on convolution neural networks and support vector machine. Int J Adv Comput Sci Appl 11(6)
111. Shams M, Elsonbaty A, ElSawy W (2020) Arabic handwritten character recognition based on convolution neural networks and support vector machine. Int J Adv Comput Sci Appl 11(8)
112. Simard PY, Steinkraus D, Platt JC (2003) Best practices for convolutional neural networks applied to visual document analysis. In: Icdar. vol. 3
113. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds) 3rd International conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings
114. Singh P, Verma A, Chaudhari NS (2016) Deep convolutional neural network classifier for handwritten devanagari character recognition. Springer, Berlin, pp 551–561
115. Slavik P, Govindaraju V (2001) Equivalence of different methods for slant and skew corrections in word recognition applications. IEEE Trans Pattern Anal Mach Intell 23(3):323–326
116. Soman ST, Nandigam A, Chakravarthy VS (2013) An efficient multiclassifier system based on convolutional neural network for offline handwritten telugu character recognition. In: 2013 National Conference on Communications (NCC). IEEE, pp 1–5
117. Sonka M, Hlavac V, Boyle R (1993) Image pre-processing. Springer US, Boston, pp 56–111
118. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res 15(56):1929–1958
119. Tato A, Nkambou R (2018) Improving adam optimizer
120. Vorontsov E, Trabelsi C, Kadoury S, Pal C (2017) On orthogonality and learning recurrent networks with long term dependencies. arXiv:1702.00071
121. Waller RA, Duncan DB (1969) A bayes rule for the symmetric multiple comparisons problem. J Am Stat Assoc 64(328):1484–1503
122. Ward R, Wu X, Bottou L (2019) Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In: International conference on machine learning. (PMLR), pp 6677–6686
123. Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85
124. Witten IH, Bell TC, Emberson H, Inglis S, Moffat A (1994) Textual image compression: Two-stage lossy/lossless encoding of textual images. Proc IEEE 82(6):878–888
125. Wright W, Caspari CP (2011) A grammar of the Arabic language. Cosimo Inc., New York

126. Wu C, Fan W, He Y, Sun J, Naoi S (2014) Handwritten character recognition by alternately trained relaxation convolutional neural network. In: 2014 14th International conference on frontiers in handwriting recognition. IEEE, pp 291–296
127. Wu H, Gu X (2015) Towards dropout training for convolutional neural networks. Neural Netw 71:1–10
128. Wu V, Manmatha R, Riseman EM (1999) Textfinder: an automatic system to detect and recognize text in images. IEEE Trans Pattern Anal Mach Intell 21(11):1224–1229
129. Xiang T, Wang J, Liao X (2007) An improved particle swarm optimizer with momentum. In: 2007 IEEE congress on evolutionary computation. IEEE, pp 3341–3345
130. Xiong H, Pandey G, Steinbach M, Kumar V (2006) Enhancing data analysis with noise removal. IEEE Trans Knowl Data Eng 18(3):304–319
131. Yang W, Jin L, Xie Z, Feng Z (2015) Improved deep convolutional neural network for online handwritten chinese character recognition using domain-specific knowledge. In: 2015 13th international conference on document analysis and recognition (ICDAR). IEEE, pp 551–555
132. Yuan A, Bai G, Jiao L, Liu Y (2012) Offline handwritten english character recognition based on convolutional neural network. In: 2012 10th IAPR International workshop on document analysis systems. IEEE, pp 125–129
133. Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv:1212.5701
134. Zhang C, Vinyals O, Munos R, Bengio S (2018) A study on overfitting in deep reinforcement learning. arXiv:1804.06893
135. Zhong Z, Jin L, Feng Z (2015) Multi-font printed chinese character recognition using multi-pooling convolutional neural network. In: 2015 13th International conference on document analysis and recognition (ICDAR). IEEE, pp 96–100
136. Zhong Z, Jin L, Xie Z (2015) High performance offline handwritten chinese character recognition using googlenet and directional feature maps. In: 2015 13th International conference on document analysis and recognition (ICDAR). IEEE, pp 846–850
137. Zhuang F et al (2020) A comprehensive survey on transfer learning. Proc IEEE :1–34
138. Zoph B, Vasudevan V, Shlens J, Le QV (2017) Learning transferable architectures for scalable image recognition. arXiv:1707.07012

Affiliations

Hossam Magdy Balaha[1] · Hesham Arafat Ali[2] · Esraa Khaled Youssef[3] · Asmaa Elsayed Elsayed[3] · Reem Adel Samak[3] · Mohammed Samy Abdelhaleem[3] · Mohammed Mosa Tolba[3] · Mahmoud Ragab Shehata[3] · Mahmoud Refa'at Mahmoud[3] · Mariam Mahmoud Abdelhameed[3] · Mostafa Mahmoud Mohammed[3]

Hesham Arafat Ali
h_arafat_ali@mans.edu.eg

[1] Assistant Lecturer at Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt

[2] Professor at Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt

[3] Graduate Students from Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt