

Received April 11, 2020, accepted April 26, 2020, date of publication May 12, 2020, date of current version May 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994248

Exploring Deep Learning Approaches to Recognize Handwritten Arabic Texts

MOHAMED ELTAY¹, ABDELMALEK ZIDOURI¹, (Senior Member, IEEE),
AND IRFAN AHMAD²

¹Department of Electrical Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

²Department of Information and Computer Science, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

Corresponding author: Abdelmalek Zidouri (malek@kfupm.edu.sa)

This work was supported by the King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

ABSTRACT Recognition of cursive handwritten Arabic text is a difficult problem because of context-sensitive character shapes, the non-uniform spacing between words and within a word, diverse placements of dots, and diacritics, and very low inter-class variation among individual classes. In this paper, we review and investigate different deep learning architectures and modeling choices for Arabic handwriting recognition. Further, we address the problem that imbalanced data sets present to deep learning systems. In order to address this issue, we are presenting a novel adaptive data-augmentation algorithm to promote class diversity. This algorithm assigns a weight to each word in the database lexicon. This weight is calculated based on the average probability of each class in a word. Experimental results on the IFN/ENIT and AHDB databases have shown that our presented approach yields state-of-the-art results.


INDEX TERMS Arabic handwriting recognition (AHR), deep learning neural network (DLNN), convolutional neural networks (CNN), connectionist temporal classification (CTC), recurrent neural network (RNN), IFN/ENIT database, long short-term memory (LSTM), bi-directional long short-term memory (BLSTM), word beam search (WBS).

I. INTRODUCTION

Optical Character Recognition (OCR) is an old field of Pattern Recognition (PR). The human reading process is the inspiration behind the development of a machine capable of reading texts with the same expertise as people. The very first OCR systems were designed to help blind people to read. Decades of scientific research have produced many practical systems ranging from the sorting of posts in post offices [1] to automatic recognition of bank checks [2], handheld scanners, and advanced systems that read the text in natural scenes [3]. The main goal of the handwriting recognition system is to convert handwritten text documents from digital image format to encoded character format documents that are readable and editable using word processing application systems. Recognition of handwriting is becoming more critical as it greatly helps to accomplish the office tasks efficiently and has solved many problems that lead to a reduction in time and effort. More than 400 million

people around the world speak Arabic [4]. Besides, Arabic letters are used in several other languages, such as Persian, Urdu, and Jawi. Compared to other languages, Arabic has not yet received much research and study. Handwritten Arabic script recognition can be achieved using either holistic or segmentation approaches. The holistic approach generally processes words into smaller components without segmentation. However, in the segmentation approach, we divide the words into characters or strokes and then pass the segments to the identifier [5]–[7]. The development of a practical cursive script segmentation algorithm is difficult and requires extensive expertise. In many types of research, handwritten text recognition algorithms that do not rely on segmentation, generally lead to significantly higher recognition accuracy [6]. To date, holistic approaches have been more successful in a limited vocabulary handwriting database [8]–[10]. However, in the case where the vocabulary of the database is very large, segmentation approaches are used.

Over the last ten years, there has been a significant advance in machine learning algorithms. Indeed, the increasing power of computer data processing has supported the analytical

The associate editor coordinating the review of this manuscript and approving it for publication was Haruna Chiroma .

capabilities of handwriting recognition systems. Deep learning methods used in the analysis of documents have been extremely successful recently. These methods are robust to deformations, scaling, and rotation. They are therefore best suited for recognition of text. Mainly, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been extensively used for text recognition. Recently, deep learning-based recognition systems have received high interest. Several studies and researches have shown that Recurrent Neural Networks (RNNs) are superior to Hidden Markov Models (HMMs) for sequence labeling tasks such as speech and online handwriting recognition [11]–[14]. One possible reason for this is that RNNs are trained in a discriminative manner, whereas HMMs are generative. Discriminative methods tend to give better results in pattern recognition tasks when a large set of data is available [15], [16]. In addition, the use of Connectionist Temporal Classification (CTC) in connection with RNNs allows recognition without prior segmentation [17]. This has made it possible to recognize offline handwritten text by using neural network-based classifiers and has thus gained popularity in recent years. Besides, the RNN's Long Short-Term Memory (LSTM) architecture allows it to capture longer contexts, which are important for the recognition of offline text tasks.

The contributions of the paper are as follows:

- 1- We present and evaluate efficient deep learning architectures used for Arabic handwriting recognition. Each system is evaluated and compared with other state-of-the-art systems.
- 2- We present various modeling options for Arabic handwriting recognition, such as (character shapes, characters, and whole-word modeling) and present the experimental results using these modeling options on a publicly available dataset.
- 3- We present a novel data augmentation algorithm to handle the imbalanced class problem in handwritten text recognition tasks.

The rest of this paper is organized as follows: Section 2 presents the related works on the recognition of Arabic text. Section 3 presents the characteristics and challenges of Arabic handwriting. Section 4 presents the different modeling options for Arabic text recognition. Section 5 presents deep learning approaches for handwriting recognition. Section 6 presents a novel data augmentation algorithm. In Section 7, we present the experiments and the results and a comparison with the state-of-the-art systems. The final section sets out conclusions and plans for future work.

II. RELATED WORKS

Al-Hajj *et al.* [18] present a two-stage system to recognize cursive handwritten Arabic words. The proposed approach is analytical without segmentation and can deal with handwriting inclination and shifted diacritical marking positions.

Dreuw *et al.* [19] proposed a continuous HMMs system to explicitly model white-spaces for Arabic handwriting recognition within different writing variants.

Graves and Schmidhuber [17] presented the Multi-Dimensional Recurrent Neural Networks (MDRNNs) for Arabic handwriting recognition. MDRNNs are a special case of directed acyclic graph networks, generalize standard RNNs by providing recurrent connections along all Spatio-temporal dimensions present in the data.

Kessentini *et al.* [20] have presented a multi-stream approach for recognizing Arabic handwritten words. The proposed method combines low-level feature streams, i.e. density-based features with different widths extracted from two different sliding windows, and contour-based features extracted from the upper and lower contours. Pechwitz *et al.* [21] described in detail the IFN/ENIT-database and presented a recognition system for the identification of Arabic words based on semi-continuous HMMs.

Rothacker *et al.* [22] focus on feature learning by estimating and applying a statistical bag-of-features model. These models are successfully used in image categorization and retrieval. Azeem and Ahmed [23] used the slanting window idea of Al-Hajj *et al.* [19] for developing three separate and continuous HMM systems. Each classification results were combined with the sum, the majority vote, and the maximum rules. Abandah *et al.* [24] used a system similar to that presented by Graves and Schmidhuber [17], but the text was explicitly segmented into graphemes before extracting features from graphemes.

In combination with a continuous HMM system, Hamdani *et al.* [25] have implemented a Bi-directional Long Short-Term Memory (BLSTM) system. Pixel gray values were used for the training of the HMM system in the first step. They were extracted from repositioned sliding windows. The next step in the training of HMMs was to apply RNN-trained features. Two successive HMM states have shared the Gaussians. Besides, tree-based decision-making contextual HMMs (tri-graph) were used. Elleuch *et al.* [26] reviewed two deep architecture systems: Deep Belief Networks (DBN) and Convolutional Deep Belief Networks (CDBN), respectively, used to recognize handwritten Arabic scripts in low-level and high-level textual images. Stahlberg and Vogel [27] proposed a novel text line image normalization procedure and a new feature extraction method using the HMMs system with deep neural network training.

Ahmad and Fink [28] proposed a multi-stage HMM-based framework with two separate systems. One to recognize Arabic core shapes after removing dots and other diacritics from text images, and another HMM system to recognize the Arabic text dots and diacritics. To recognize the core shapes of the Arabic text the core-shape recognition system was used. The Arabic words were finally recognized by incorporating information from the diacritics recognition system. Rabi *et al.* [29] proposed a cursive handwritten Arabic text recognition system based on Hidden Models (HMM). This system is analytical and uses embedded training to perform and improve the character models without explicit segmentation.

Almodfer *et al.* [30] improved AlexNet’s recognition of handwritten Arabic words by adopting a drop-out regularization that prevented the proposed system from overfitting problems and reduced word error rates. Besides, they investigated the performance of ReLU and *tanh* activation functions in the fully connected layers. Ahmad and Fink [31] presented a new way of representing Arabic characters by separating the core shapes from the diacritics and then representing these core shapes through smaller units called sub-core shapes.

III. BACKGROUND

Arabic is an ancient Semitic language. It is the lingua franca of the Arab world. Arabic is ranked among the top six major languages in the world [32]. Most of the Arabic characters are connected while they are written. Arabic characters are written from right to left. Each character can have up to four distinct shapes depending on its position in the word. Table 1 shows the basic Arabic characters which are usually used in all Arabic scripts.

TABLE 1. The basic Arabic characters.

أ	ب	ت	ث
ح	خ	د	ذ
ر	ز	س	س
ك	ط	ظ	ع
ل	ف	ق	ن
م	و	ي	ة

In comparison to Latin scripts, Arabic scripts handwritten and computer-printed are highly comparative. Although character shapes have slight differences due to the style of handwritten text, the similarity between characters is generally the same. Fig.1 shows a sample of handwritten and computer-printed Arabic texts.

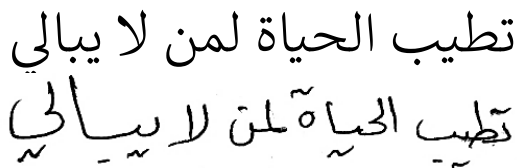


FIGURE 1. Computer printed and handwritten versions of the same Arabic text.

A. CHARACTERISTICS AND CHALLENGES OF ARABIC HANDWRITING

Handwritten Arabic text recognition is considered a difficult task compared to Latin script. This is due to its cursive nature, presence of diacritics, diagonal strokes, change in the shape of characters according to their location in the word, inter-word, and intra-word spaces, and difficulty in segmentation. Besides, dots play a significant role in Arabic characters.

Some of the shapes of the characters are very similar, but the distinction is made between the characters by the position and the number of dots that can be either above or below the character body. For instance, the three characters, (Thaa ث), (Taa ت) and (Baa ب) have the same main body shape but differ only by the number of dots and their positions.

Besides, Arabic utilizes diacritical imprints (vocalized content) to control the articulation of words, see Fig.2. However, these are seldom showing up in handwritten documents. They are mostly used in didactic documents or when the context is ambiguous.

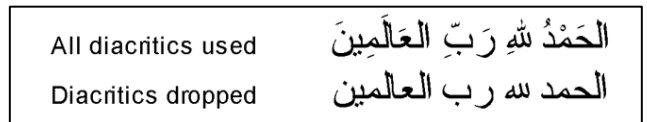


FIGURE 2. Utilization of diacritics in Arabic script.

Arabic script is cursive, meaning that a word’s characters are linked on an imaginary horizontal line called a baseline. There are also strokes above and below the baseline referred to as ascenders and descenders, as shown in Fig.3.

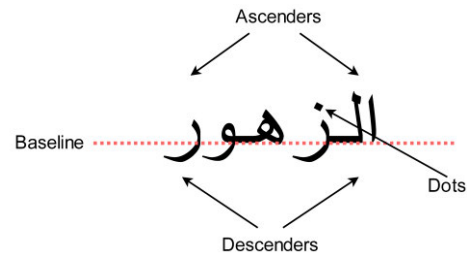


FIGURE 3. Baseline, Ascenders, and Descenders, as shown in a word.

Furthermore, there are six characters (ا, د, ذ, ر, ز, و) which do not connect in a word to a subsequent character, and this causes the word to be separated into parts, as shown in Fig.4. These parts are called sub-words or Pieces of Arabic Word (PAW).

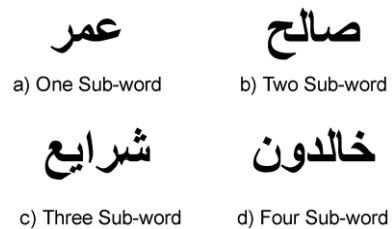


FIGURE 4. Words with different sub-words.

Moreover, when two or more characters are joined as a single glyph, a ligature occurs. The use of ligature in Arabic is common. e.g. (a) “laam-heh (ح),” (b) “laam-meem (م),” and (c) “laam-Alif (لا).” Fig.5 shows cases of associated characters.

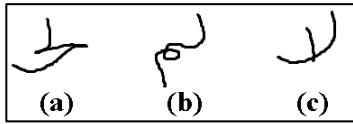


FIGURE 5. (a) Laam-heh, (b) laam-meem, and (c) laam-alef ligatures.

The Arabic handwritten text presents many challenges to the Optical Character Recognition (OCR) developer that can be presented as follows:

1. Arabic word comprises at least one associated segment (sub-word), and each one contains at least one character that can be covered with different characters or diacritics. Besides, different characters can be consolidated vertically to shape a ligature. (Fig.6a)
2. Some words contain touching, interlaced, or broken characters. (Fig.6b)
3. Each writer has an individual handwriting style. (Fig.6c)
4. Some Arabic characters have diacritics (a diacritic might be set above or beneath the body of the character). These diacritics can also be overlapping. (Fig.6d)

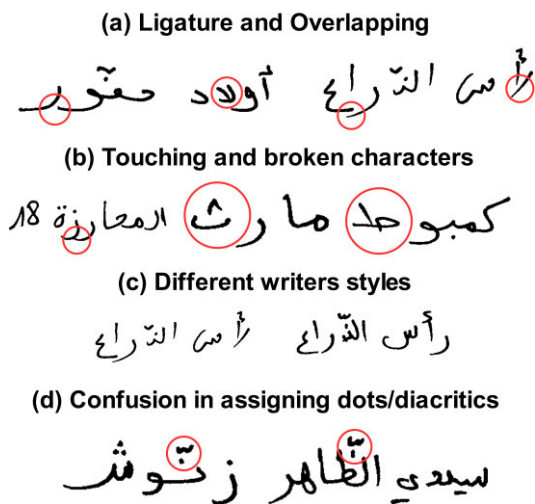


FIGURE 6. Some challenges in handwritten Arabic text recognition.

B. HANDWRITING RECOGNITION SYSTEMS

In general, character recognition systems can be divided into two types, as shown in Fig.7: online and offline. The online character recognition is performed during the writing process. Usually, this technique uses equipment such as a special pen and tablet, which uses the digitized trace of the pen to recognize characters. Pre-written records cannot, therefore, be recognized. The offline character recognition, on the other hand, is concerned with images scanned from previously written documents. The off-line recognition of texts can also be divided into two categories: the recognition of handwritten and printed characters.

Printed characters are uniform in shape and structure, while handwritten characters have different styles and sizes for both

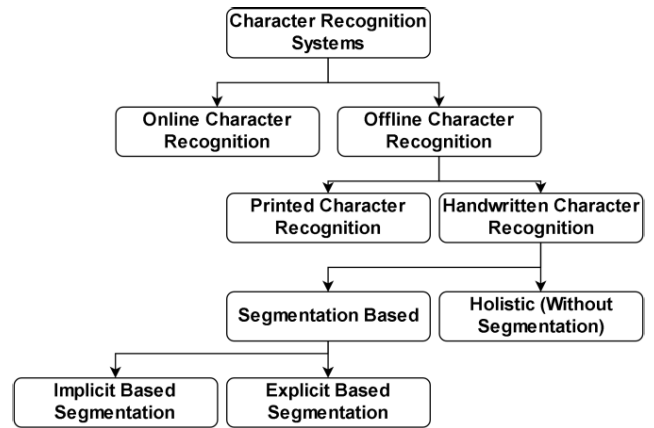


FIGURE 7. Different modalities of character recognition.

the same writer and different writers. Handwritten words can be recognized in two ways: whole word recognition without segmentation or segmentation-based recognition.

Although the segmentation process is the primary source of error recognition, most systems avoid this step and recognize the word as a whole but if the texts presented in the image are not valid words in the database or the in case of open vocabulary recognition, holistic methods will fail, and we need to rely on segmentation approaches.

Segmentation-based approaches are broadly divided into two categories:

1. Implicit segmentation or internal segmentation in which segmentation and identification of character are achieved simultaneously [33]. The implicit segmentation approach allows the recognizer to decide the best hypothesis for segmentation.
2. Explicit segmentation or external segmentation where an image of a sequence of characters is divided into sub-images of individual characters then classified [34]. Because of the presence of ligature and the cursive nature of the Arabic script, several researchers introduced techniques based on the recognition of the entire word without segmentation [17], [26].

Generally, recognition of any handwritten script involves several steps, as shown in Fig.8. The first step is to improve the readability of the text image and to remove any unwanted details (Image Pre-Processing). The pre-processing step usually involves many tasks such as binarization, noise removal, baseline detection, and normalization. Next, we extract features from handwritten texts. Typically, the feature vector contains the characteristics of the given character or word. In the final step, we try to identify the given character or word by comparing its characteristics to one of the classes in the database, assuming that the training classes and the training model have been provided.

IV. ARABIC HANDWRITING RECOGNITION MODELING OPTIONS

In this section, we will present the various modeling options that we have explored during this work. In the next three

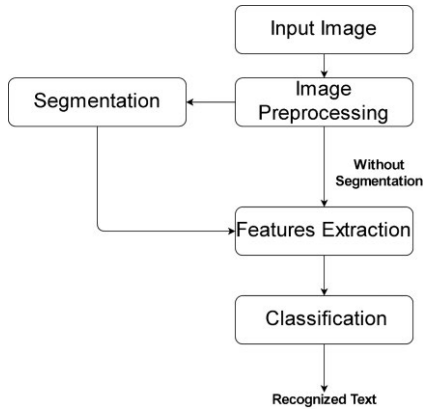


FIGURE 8. General steps for handwritten text recognition.

sub-sections, we will present the three different but related modeling approaches that we have investigated. We, thus, describe three broad options of Arabic handwriting modeling approaches, these modeling options are similar to the acoustic speech recognition model [14]. The experiments and the results related to these modeling and training options are presented in section 7.

A. WHOLE-WORD MODELING (HOLISTIC RECOGNITION)

Arabic words or PAWs were used by some researchers as modeling units long ago. There are many problems with this approach. One such problem is that it usually leads to a massive model set as each word will be represented by a separate class. There may be hundreds or thousands or more words in a lexicon. This means that more training data are required as each word should have enough samples in the training set to train model parameters adequately. It may only be appropriate for text recognition tasks with a very small lexicon size. [16], [21].

B. USING ARABIC CHARACTERS AS MODELING UNITS

In this modeling option, the basic Arabic characters (28 characters, ten digits, and seven ligatures) are used as modeling units. However, because Arabic has many position-dependent models, modeling Arabic handwriting, using characters as models are not the best way, although it was used in the early days of Arabic text recognition. Many Arabic character shapes look quite different visually and mapping them to one model may not be the best option.

C. USING THE CHARACTER SHAPES AS MODELING UNITS

The use of Arabic character shapes as distinct models is the most successful and widely used approach [25]. Table 2 shows some Arabic characters, along with their position dependent shapes.

Although this approach generally works well, it still has a problem. The number of modeling units rises four times from 28 to about 100, with the exclusion of numbers and special characters. This creates a big model set with a large recognizer. Moreover, a wide range of modeling units

TABLE 2. Names of some Arabic characters and their position dependent shapes.

Character's Name	Alone (Isolated)	Ending	Middle	Beginning
Bā'	ب	ـب	ـبـ	بـ
Tā'	ت	ـت	ـتـ	تـ
Thā'	ث	ـث	ـثـ	ثـ
Jīm	ج	ـج	ـجـ	جـ
Hā'	ح	ـح	ـحـ	حـ
Khā'	خ	ـخ	ـخـ	خـ
Dāl	د	ـد	ـدـ	دـ
Dhāl	ذ	ـذ	ـذـ	ذـ
Rā'	ر	ـر	ـرـ	رـ

requires extensive training data to ensure that each model is properly trained.

V. DEEP LEARNING APPROACHES

Deep models for Arabic handwriting recognition are generally rare compared to other languages because it is challenging to deal with cursive writing [35]. Deep networks are specifically designed to simulate human brain activity, pattern recognition, and input through different layers of simulated neural connections. The most straightforward deep neural network consists of an input layer, an output layer, and a hidden layer between them. Each layer performs specific types of sorting and ordering.

Due to the success of modern deep neural network architectures, it is envisaged that state-of-the-art handwriting recognition systems will be either hybrid systems (deep networks with some segmentation and features extraction) or pure neural recognizers with deep architectures [36]. There are many deep learning systems and techniques used for handwriting recognition. Some of these systems and architectures are presented in the following subsections.

A. CONVOLUTIONAL NEURAL NETWORKS (CNNs)

The Convolutional Neural Network (CNN) is a class of artificial deep feed-forward neural networks, usually used for image classification. CNN has several layers that can efficiently learn high-level features from labeled training data. In fact, its key idea is to use convolutional and pooling layers to gradually extract more abstract patterns that can be carried out locally [37].

CNN is like other deep neural networks consist of an input layer, an output layer, and many hidden layers in between. These layers perform data-specific operations to learn features. The four most commonly used layers are convolution, Rectified Linear Unit (ReLU), pooling, and fully connected.

The convolution layer puts the input images through a set of convolution filters; each filter will extract a specific feature from the image.

The rectified linear unit (ReLU) allows for fast and better training by mapping negative values into zero and maintaining positive values. This is often called activation since only features activated is transferred to the next layer.

The pooling layer simplifies output through nonlinear down sampling and reduces the number of parameters needed to be learned by the network. This operation is repeated over 10 or 100 layers, which allows the identification of different characteristics on each layer.

The input feature vector is represented by the fully connected layer. This feature vector holds vital input information. When the network is trained, this feature vector is then further used for classification, regression, or input into another network such as RNN to be translated into another output type, etc. It also serves as an encoded vector. This feature vector is used during training to determine the loss and to help the network get trained.

A popular example of CNN architecture is the AlexNet architecture shown in Fig.9. The AlexNet initially proposed by Krizhevsky et al. [38], is consisted of eight layers; the first five layers are convolutional layers, some of them followed by max-pooling layers, and the last three layers are fully connected layers. It uses the non-saturating activation function “ReLU” that showed improved training performance over *tanh* and sigmoid.

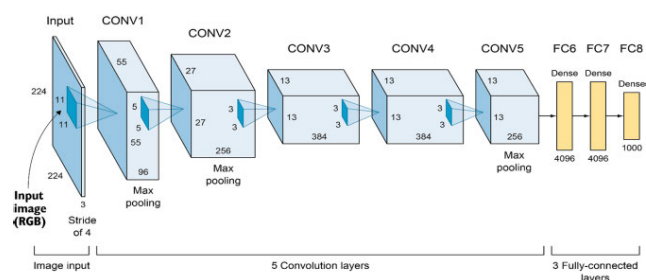


FIGURE 9. The AlexNet [38] architecture.

B. RECURRENT NEURAL NETWORKS (RNNs)

A Recurrent Neural Network (RNN) is a robust deep neural network, commonly used to solve problems of machine learning involving sequence inputs. This particular ability enables RNN to learn the context of labels, as the word’s characters cannot be regarded as genuinely independent components, which is a great advantage in handwriting recognition.

Although the traditional RNN gives the network a memory, its ability to preserve old contents is still quite limited due to the vanishing gradient problem [39]. Usually, it is difficult for an RNN to bridge gaps between the presentation of the relevant input and the relative target events over more than 10-time steps. Because of this, the Long Short-Term Memory (LSTM) [40] model is used to overcome this problem. RNNs architecture involving (LSTM) enables it to capture longer contexts which may be important for offline text recognition tasks.

RNNs, are capable of modeling sequential data for sequence recognition and prediction. In traditional neural networks, the input and output data are assumed to be independent of each other. In many applications, using this assumption is not a good idea. A simple RNN has three layers,

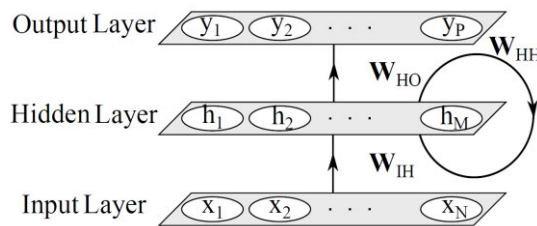


FIGURE 10. A folded recurrent neural network [41].

as shown in Fig.10, which are input, recurrent, and output layers [41]. There are N input units in the input layer. This layer’s inputs are a sequence of vectors over time t such as $\{\dots, x_{t-1}, x_t, x_{t+1}, \dots\}$ where $x_t = \{x_1, x_2, x_3, \dots, x_N\}$.

The input units in a fully connected RNN are connected to the hidden units in the hidden layer where a weight matrix W_{IH} is used to define the connections. The hidden layer has M hidden units $h_t = (h_1, h_2, \dots, h_M)$ connected over time with recurrent connections [41] see Fig.11.

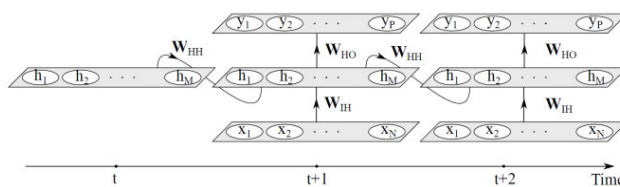


FIGURE 11. Unfolded recurrent neural network through time [41].

The initialization of hidden units with small non-zero elements may improve overall network performance and stability [15]. The hidden layer defines the system’s state space or “memory” as

$$h_t = fH(o_t), \tag{1}$$

where

$$o_t = W_{IH}x_t + W_{HH}h_{t-1} + b_h, \tag{2}$$

$fH(\cdot)$ is the activation function of the hidden layer, and b_h is the hidden units’ bias vector. The hidden units are connected with weighted W_{HO} connections to the output layer. The output layer has P units $y_t = (y_1, y_2, \dots, y_P)$ that is computed as

$$y_t = f_o(W_{HO}h_t + b_o) \tag{3}$$

where, f_o is the activation functions and b_o is the bias vector in the output layer.

As the input-target pairs are sequential over time, the above steps are repeated over time $t = (1, \dots, T)$.

The equations (1) and (3) show that RNN consists of absolute non-linear equations of state that can be iterated over time. The hidden states provide a prediction in each time step based on the input vector at the output layer.

The hidden state of an RNN is a set of values that, apart from the effect of any external factors, over several time steps,

summarizes all the unique information needed about the network’s past states. This integrated information can define the network’s future behavior and predict accurately at the output layer [42]. In each unit, an RNN uses a simple nonlinear activation function. However, such a simple structure is capable of modeling rich dynamics if it is well trained in time steps.

C. CONNECTIONIST TEMPORAL CLASSIFICATION (CTC) LAYER

In the context of RNN, the use of Connectionist Temporal Classification (CTC) enables recognition without prior segmentation [17]. This has enabled the use of neural network classifiers for offline recognition of handwritten texts, which has become more popular in recent years [17], [24], [25].

The segmentation process for Arabic text, printed or handwritten, is a highly error-prone task. The Connectionist Temporal Classification (CTC) layer can transcribe the data without prior segmentation. Graves *et al.* [11] introduced the approach of CTC which was initially designed to recognize speech [14], and then the idea was extended to recognize handwriting [17]. The CTC layer predicts the transcription for the test images in the recognition phase. The out-transcriptions are matched with the ground truth of the line image using Levenshtein edit distance.

By interpreting the network output as a distributed probability over all possible label sequences on the given input sequence, CTC allows for the previously mentioned direct alignment between the input variables and target label. The last layer in the network has $N+1$ outputs, where N is the total number of labels. These outputs specify the probability that each label is observed or not observed at a given time. With the use of single-character probabilities, a set of probabilities can be obtained that possible match outputs given to an input one. This logic can be described by equation (4).

$$p(\pi | x) = \prod_{t=a}^T y_{\pi_t}^t, \quad \forall \pi \tag{4}$$

where:

- x : The input sequence.
- π : A possible output sequence.
- $y_{\pi_t}^t$: The probability of a given label from π at time t

A special β operator is defined for the correspondence between previous steps π and the final output sequence l through the removal of blank labels. As more than one sequence of π can be a single final sequence, the probability of a specific final sequence of l will be calculated as follows:

$$p(l | x) = \sum_{\beta(\pi)=l} p(\pi | x) \tag{5}$$

The maximum probability $p(l|x)$ is considered to be the final output for each given input x and sequence l . For handwriting recognition, a word image is considered a time sequence, in which time is modeled as one unit of time along the width of the image with a one-pixel slice.

Following preprocessing, pixel values have been used as network input features. The general recognizer structure is displayed in Fig.12.

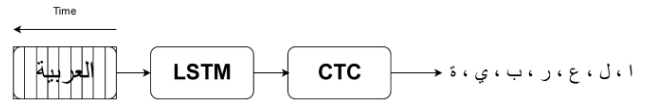


FIGURE 12. CTC output layer connected to the LSTM network.

D. TOKEN PASSING DECODER

Graves *et al.* [43] introduced the token passing algorithm which restricts its output to a dictionary. The token passing decoder solves the problem of word decoding, i.e. the most likely word in the input sequence, where the word output is limited by vocabulary. For single words, the token passing decoder expects a sequence of letter probabilities of length t as provided by the neural network, along with a word w as a sequence of ASCII characters, and returns a corresponding score, i.e. the probability that the input to the neural network was indeed the given word [14]. This means that, the probability $p(l | x) = \sum_{\beta(\pi)=l} p(\pi | x)$ in (5) could be approximated by:

$$p(w | x) = \max_{\pi, B(\pi)=w} p(\pi) \tag{6}$$

E. WORD BEAM SEARCH (WBS) DECODER

Word Beam Search (WBS) decoding is an algorithm used to decode the output matrix of the CTC layer. The WBS decoder is placed just following the CTC layers for output decoding. The main advantages of the WBS decoder [44] over token passing decoder are:

- It is faster than token passing.
- It allows an arbitrary number of non-word characters between words (numbers, punctuation marks).
- Words constrained by dictionary.

Scheidl *et al.* [44] presented a WBS decoder which is a modification of the Vanilla Beam Search (VBS) decoding algorithm [44] that has the following characteristics:

- Words are limited to words in the dictionary.
- Any non-word character number between words is permitted.
- Optionally, we can integrate an LM word-level bigram.
- A better runtime (in terms of time-complexity and real-time on a computer) than token passing.

The iterative beam search decoding creates and scores the text (beams). Algorithm 1 shows the pseudo-code for a basic-version: the beam list is initialized with an empty beam (line.1) and the corresponding scoring score (line.2). The algorithm then iterates the neural network output matrix in all-time stages (line.3 to 15).

The beam search decoding algorithm allows arbitrary character strings, which are required to decode numbers and punctuation marks while token-passing limits its output to words in the dictionary to avoid spelling mistakes. Of course, all words to be recognized in the dictionary must be included.

Algorithm 1 Pseudo-Code for a Basic Word Beam Searching [44]

```

Input: NN output matrix mat, BW
Result: decode text
1 beams = {∅}
2 score {∅, 0} = 1
3 for t = 1 . . . T do
4     estBeams = bestBeams(beams, BW);
5     beams = {};
6     for b ∈ bestBeams do
7         beams = beams ∪ b;
8         scores(b, t) = calcScore(mat, b, t);
9         for c ∈ alphabet do
10            b' = b + c;
11            scores(b', t) = calcScore(mat, b', t);
12            beams = beams ∪ b';
13        end
14    end
15 end
16 return bestBeams (beams, 1);
    
```

F. BI-DIRECTIONAL LONG SHORT-TERM MEMORY (BLSTM)

The Bi-directional Long Short-Term Memory (BLSTM) architecture was introduced by Graves *et al.* [17], [43] to overcome the limitations of LSTM. One of the main drawbacks of RNN and LSTM is that they are processing information just based on the previous context.

In many applications, including speech recognition, the Bi-directional LSTM (BLSTM) structure is preferred. This structure can be trained using all available input information in the past and future of a specific time frame. BLSTMs contain two separate hidden layers, one processes the input sequence forwards (positive time direction), while the other processes it backward (negative time direction).

Both hidden layers are connected to the same output layer, providing it with access to the past and future context of every point in the sequence. BLSTM outperforms unidirectional LSTMs and standard RNNs, and it is also much faster and more accurate. Fig.13 shows the architecture of a BLSTM network.

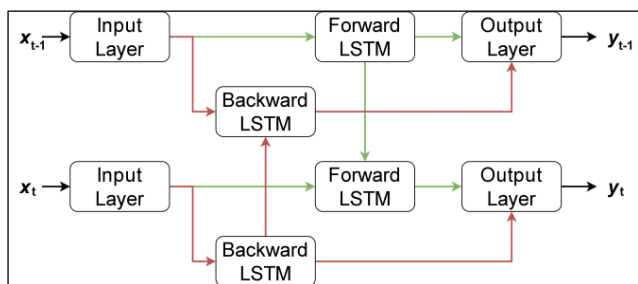


FIGURE 13. The architecture of a bi-directional recurrent neural network.

VI. A NOVEL ADAPTIVE DATA AUGMENTATION ALGORITHM

Imbalanced classifications pose challenges as they lead to systems with poor predictive performance, especially for the minority class. A neural network that is trained on a database with imbalanced classes will bias its decision towards the prevalent classes in the set [45]. In the presence of limited training samples, data augmentation is commonly used in many deep learning approaches. Augmented data using standard augmentation techniques e.g. (rotation, position shifting, zoom, shear, etc.) does not affect the distribution of labels in the original dataset. It means that if we have imbalanced data, the data will continue to be imbalanced after data augmentation.

The issue of class imbalance based on the difference in character frequencies is not specific to any language or dataset. It is a common observation for texts in natural languages. For example, characters ‘e’, ‘t’, and ‘a’ are the most frequent characters whereas characters ‘z’, ‘q’, and ‘x’ are the least frequent characters in the English language [46]. Similarly, for Arabic, based on text analysis of an Arabic corpus containing more than five million characters [47], it was found that not only the characters are unevenly distributed but also the distribution is highly skewed. The two most common characters, i.e., alif (ا) and lām (ل), constituted almost a quarter of all the texts. On the other hand, the five least occurring characters (out of the total of 28 characters) constitute only around 2% of the entire text. The frequency of character zā’ (ظ) was almost 70 times lower than the frequency of character alif (ا).

Now, if we consider the character shapes distribution rather than characters, the distribution will be even more skewed. There will be very few occurrences in some character shapes from the low-frequency characters and thus there is the issue of adequately training them. In this section, we present a novel data augmentation algorithm that can balance the distribution of the character samples in a dataset.

First, we need to know the size of the database lexicon (L) and the number of modeling units or classes (n). For example, if we use Arabic characters as modeling units, in this case, we will have 45 models (28 basic letters, 10 digits, and 7 ligatures) (n=45).

Second, we assign a weight (W_i) to each word in the lexicon. This weight is calculated for each word based on the sum of the inverse average probability of the characters that make up that word. The inverse probability was used because we are interested in words composed of infrequent characters. Therefore, words with a high weight are composed of characters that infrequently appear in training data, while words with a low weight value are composed of characters frequently seen in training data.

In general,

$$W_i = \frac{\sum_1^N \frac{1}{p_n}}{N} \tag{7}$$

where $i = \{1, \dots, L\}$, N is the total number of characters in the word i and p_n is the probability of each modeling unit (class) in the training data sets.

$$p_n = \frac{\text{Total Number of class}(n)\text{occurrence}}{\text{Total Number of all classes occurrence}} \times 100 \quad (8)$$

After finding all the weights, we normalize them by dividing each of the weights by the sum of all the weights. Therefore, normalized weights are like probabilities:

$$W_{i,normalized} = \frac{W_i}{\sum W_i}, \text{ where } i = \{1, \dots, L\}; \quad (9)$$

Now we can use these to augment new samples to the original database e.g. if we want to augment M new samples to the original database then we can calculate the number of augmented samples needed for each word image by:

$$\begin{aligned} [W_{i,normalized} \times M] \\ = \text{Total No. of samples augmented for class } i. \end{aligned}$$

By applying this to all the words in the lexicon, we will eventually balance the database, because we augment new data based on the inverse average probability of occurrence of each class. Words composed of infrequent characters will be augmented by a greater number compared to words composed from high frequent characters. Algorithm 2 shows the pseudo-code of our data-augmentation technique.

Algorithm 2 Pseudo-Code for Adaptive Data Augmentation Technique

- step 1:** Determine the lexicon size of the given database (L) and the number of modeling units (n).
step 2: For each word (i) in the lexicon, calculate the inverse probability of each model ($1/p_n$).
step 3: For each word (i) in the lexicon,

$$W_i = \frac{\sum_1^N \frac{1}{p_n}}{N} \text{ where } i = \{1, \dots, L\};$$

- step 4:** For each W_i , we normalize it by:

$$W_{i,normalized} = \frac{W_i}{\sum W_i} \text{ where } i = \{1, \dots, L\};$$

- step 5:** Determine the number of new augmented data (M).
step 6: For each word (i) in the lexicon,
 No. of samples augmented = $M \times W_{i,normalized}$
-

VII. EXPERIMENTS AND RESULTS

In the next subsections, we present the IFN/ENIT database, AHDB database, data preprocessing, experiment configurations, results of the evaluation, and the comparison with the state-of-the-art systems.

A. IFN/ENIT DATABASE

The IFN/ENIT database is the most widely used and popular database for handwritten Arabic text recognition

research published by Pechwitz and Maergner [21]. It consists of 1000 forms, written by 411 distinct writers and it is divided into five training and testing groups and it is freely available for research purposes. It includes approximately 27,000 handwritten Arabic names representing Tunisian city names. The size of the lexicon is 937 place names. A Ground Truth (GT) file has been compiled for every word in the database. This file contains information on the word, such as the baseline position and information about the individual characters used. Fig. 14 shows some sample images from the IFN/ENIT database.



FIGURE 14. Samples from the IFN/ENIT database.

B. AHDB DATABASE

The Arabic handwritten database (AHDB) developed by Al-Ma'adeed *et al.* [48] includes approximately 10,081 handwritten Arabic words representing numbers and quantities used in cheques, as well as the most popular words in Arabic writing. The size of the lexicon is 96 of which 67 are handwritten word numbers that can be used in handwritten cheques. The remaining 29 words represent the most popular words in Arabic. Fig.15 shows some sample images from the AHDB database.



FIGURE 15. Samples from the AHDB database.

C. PREPROCESSING

In any recognition system, the pre-processing of the text images is an important step. The primary objective of this step is to improve the readability of the text images and to remove details that do not have any discriminative power during the recognition process. The images were already binarized by the authors [21], [48], and the only steps we took in the pre-processing were:

1. For the LSTM network, we resized all images to a fixed height of 96 pixels, and the width was chosen such that each image keeps its original aspect ratio. This step is essential in our system to ensure constant vector dimensions because LSTM expects a fixed-size vector.

2. The depth of the proposed deep learning networks is higher than traditional classifiers because of the various hidden layers. In this case, we face this serious problem because we have limited data. To overcome this issue, we used standard data augmentation techniques. In the first part of the experiments, the augmentation techniques were selected such as:

(i) For each image in the database, we have generated 8 additional augmented images by rotating images by a small angle from -3 to 3 degrees and adding those images to our training sets. See Fig.16.

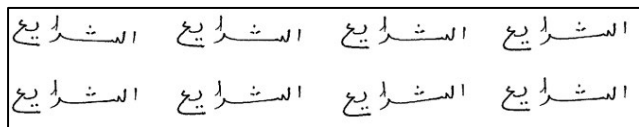


FIGURE 16. A word taken from the database rotated at different angles: (-3 to 3 degrees).

(ii) For each of the original images in the database, we generated five additional augmented images using shear transformation. See Fig.17.

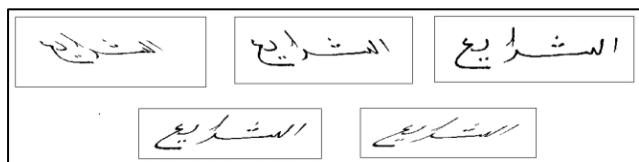


FIGURE 17. Shearing a word in the database with different shearing factor.

In Sec.7.8 we re-evaluated the performance of all systems by using our adaptive data augmentation explained in Section 6 instead of using these standard augmentation techniques.

D. EXPERIMENT CONFIGURATIONS

We have compared the performance of the RNN-LSTM network to CNN (AlexNet [38]) holistically. The AlexNet network consisted of five convolutional layers, a max-pooling layer, a dropout layer, and three fully connected layers. After each convolutional and fully connected layer, ReLU is applied. Before the first and second fully connected layer dropout is applied. The receptive field of each convolution layer is 3×3 . The fifth convolution layer is followed by three fully connected layers, with the first two having 512 channels each. The AlexNet was originally designed to identify 1000 classes. We have replaced the last fully connected layer by a new fully-connected layer with as many outputs as the number of classes (937 for the IFN/ENIT database and 96 for the AHDB database). Due to the high training time associated with such a large network, we have used transfer learning [49].

The architecture of the RNN-LSTM network for classification was chosen such that it starts with a sequence input

layer followed by an LSTM layer. The LSTM layer is followed by a fully connected layer to predict class labels, a SoftMax layer, and a classification output layer. The SoftMax layer is connected to the CTC-token passing decoder. Later, the CTC layer is modified to adopt the word beam search algorithm [44]. The block diagram of the RNN-LSTM network is shown in Fig.18.

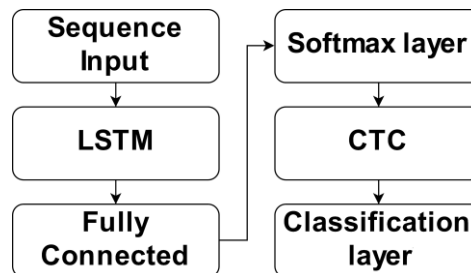


FIGURE 18. Block diagram of the LSTM network.

For the BLSTM network experimented in this paper, we have used the Recurrent Neural Network Library (RNNLIB) for sequence learning problems which is forked from Alex Graves’s work [50]. This library provides 1D and multidimensional LSTM network with ground truth alignment using the CTC decoder. The proposed network has 200 extended blocks of LSTM memory in each of the forward and backward layers. There is one memory cell, an input gate, an output gate, a forget gate, and three peephole connections for each memory block. The activation functions for the cell input and output are tanh while the activation function for the gate is a logistic sigmoid. All the CTC systems have used a decaying learning rate. The initial learning rate was set at 0.01 and gradually reduced to 0.0001.

For consistency, identical training parameters were utilized by setting the learning rate to 0.0001. It also provides a heuristic decoding process for mapping the output of the network to a sequence of symbols. For all the experiments reported in this paper, the IFN/ENIT database and AHDB were used to train and test all the systems. MATLAB [51] was used to pre-process the images, and experimental models were implemented using TensorFlow [52] and OpenCV [53].

E. PERFORMANCE EVALUATION

In this section, we present the experiments we conducted, the results we obtained, and the resulting discussions of all structures presented previously. The performance of all handwriting recognition systems was evaluated using two metrics.

1. Word Accuracy Rate (WAR): Word percentage correctly recognized with all characters.

$$WAR (\%) = \left[1 - \frac{S + I + D}{N} \right] \times 100 \tag{10}$$

where:

- S, is the total number of substituted words.
- I, is the total number of inserted words.
- D, is the total number of deleted words.
- N, is the total number of words in the evaluation set.

After aligning the recognized word string against the transcription, the number of words replaced, inserted, and deleted is counted. Alignment is known as the problem of maximum substring matching [28].

2. Character Accuracy Rate (CAR): The percentage of characters that have been correctly recognized in all words. This rate is usually measured using the edit distance method. It is defined as the minimum number of insertions, deletions, or substitutions of a single terminal needed to transform other of the strings into the other one.

$$CAR (%) = \left[1 - \frac{S + I + D}{N} \right] \times 100 \quad (11)$$

where:

S, is the total number of substituted characters.

I, is the total number of inserted characters.

D, is the total number of deleted characters.

N, is the total number of characters in the evaluation set.

After aligning the recognized character string against the transcription, the number of characters substituted, inserted, and deleted is counted [25].

F. RESULTS OF HOLISTIC HANDWRITING RECOGNITION. (WHOLE-WORD MODELING)

Table 3 shows the results of AlexNet and RNN’s holistic recognition on the IFN/ENIT and AHDB databases. Both of the networks were trained using the same configuration described in Sec.7.1 to evaluate their performances similarly. Both of the systems were trained and tested using similar train-test configurations and both of the networks were trained for 1,000 epochs.

TABLE 3. AlexNet vs. RNN (Holistic Recognition).

Database	Train-test configurations	ALEXNET	RNN (LSTM)	ALEXNET	RNN (LSTM)
	Augmented Data (Standard)	×	×	√	√
		WAR %	WAR %	WAR %	WAR %
IFN/ENIT	<i>abc-d</i>	90.29%	89.13%	95.27%	95.77%
	<i>bcd-a</i>	86.34%	85.66%	92.17%	93.29%
	<i>abcd-e</i>	83.12%	82.16%	89.39%	90.16%
	<i>abcde-f</i>	71.43%	73.67%	83.92%	81.66%
AHDB	70% training 20% testing 10% validation	94.97%	93.30%	96.80%	96.69%

The overall difference in performance between the two networks was negligible. The AlexNet network was able to achieve 90.29% compared to 89.13% WAR on the IFN/ENIT database without using any augmented data. On the AHDB database was able to achieve 92.97% compared to 91.30% WAR without using any augmented data.

Conversely, standard neural augmentation performs remarkably better than no augmentation. In AlexNet, the neural augmentation performed 95.27% to 90.29% compared to no augmentation on the IFN/ENIT database while it

performed 96.80% to 92.97% on the AHDB database. the RNN with neural augmentation performed the second-best, where the word accuracy rate increased from 89.13% to 95.77%. on the IFN/ENIT database while it performed 96.69% to 92.97% on the AHDB database.

G. RESULTS OF ARABIC HANDWRITING RECOGNITION USING DIFFERENT MODELING OPTIONS

1) RESULTS OF ARABIC HANDWRITING RECOGNITION USING CHARACTER SHAPES AS MODELS

In this section, we present the results obtained for word recognition tasks using character shapes as modeling units. The IFN/ ENIT database contains the transcriptions for the word image at the character-shape level and we have created similar transcriptions for the AHDB database. We ended up with a total of 157 models in our recognition system. We have used the LSTM-CTC architecture with the Token passing and WBS decoders explained in sections 5.4 and 5.5. Table 4 shows the accuracy rate of each system without using any augmented data, and Table 5 shows the accuracy rate when using augmented data. The results show that by using standard data augmentation techniques, the accuracy rate is improved. The maximum CAR obtained on the IFN/ENIT database is 97.17% while the maximum WAR on the IFN/ENIT database is 96.22%. The maximum CAR obtained on the AHDB database is 96.77 while the maximum WAR is 95.99 %. Clearly, by using the CTC layer with the WBS decoder, the network gives much better results.

TABLE 4. Accuracy Rates of the Systems using character shapes as models (RNN-LSTM) – (No augmented data).

Database	Train-test configuration	LSTM-CTC (Token Passing)		LSTM-CTC (WBS-Decoder)	
		CAR %	WAR %	CAR %	WAR %
IFN/ENIT	<i>abc-d</i>	95.23%	94.12%	96.17%	94.83%
	<i>bcd-a</i>	90.14%	89.97%	93.42%	92.72%
	<i>abcd-e</i>	88.29%	86.13%	89.94%	88.91%
	<i>abcde-f</i>	77.41%	75.17%	79.12%	81.86%
AHDB	70% training 20% testing 10% validation	91.27%	90.21%	92.89%	93.49%

A further improvement to the current system can be achieved if we replace the LSTM layer by the Bi-directional LSTM (BLSTM) layer. The use of BLSTM can further improve the network. In BLSTM, the forward layer learns the context of the input by processing the sequence from input to output, while the backward layer performs the opposite operation by processing the sequence from output to input. Table 6 shows the results of using the BLSTM network. We can see that the accuracy rates have improved significantly compared to LSTM. In sporadic cases when using BLSTM the improvement of the results can be very limited, this can be explained in Lee et al. [58] by the prevalence of short words within the dataset but in general, replacing the LSTM layer with Bi-directional Long Short-Term Memory (BLSTM) enables higher recognition rates to be achieved.

TABLE 5. Accuracy Rates of the Systems using character shapes as models (RNN-LSTM) – (using Standard data augmentation).

Database	Train-test configuration	LSTM-CTC (Token Passing)		LSTM-CTC (WBS-Decoder)	
		CAR	WAR	CAR	WAR
		%	%	%	%
IFN/ENIT	<i>abc-d</i>	96.03%	95.39%	97.17%	96.22%
	<i>bcd-a</i>	93.16%	92.29%	94.18%	93.17%
	<i>abcd-e</i>	90.43%	88.19%	91.24%	90.14%
	<i>abcde-f</i>	87.33%	85.29%	89.14%	87.94%
AHDB	70% training 20% testing 10% validation	94.57%	94.11%	96.77%	95.99%

TABLE 6. Accuracy Rates of the Systems using character shapes as models (BLSTM) – (using Standard data augmentation).

Database	Train-test configuration	BLSTM-CTC (Token Passing)		BLSTM-CTC-WBS (Word Beam Search)	
		CAR	WAR	CAR	WAR
		%	%	%	%
IFN/ENIT	<i>abc-d</i>	97.23%	96.82%	98.16%	97.70%
	<i>bcd-a</i>	95.16%	94.19%	94.98%	93.20%
	<i>abcd-e</i>	92.13%	91.95%	94.14%	93.75%
	<i>abcde-f</i>	91.23%	90.20%	92.33%	91.92%
AHDB	70% training 20% testing 10% validation	96.72%	95.41%	97.47%	97.09%

2) RESULTS OF ARABIC HANDWRITING RECOGNITION USING CHARACTERS AS MODELS

In this section, we shall present the results of Arabic handwriting recognition using Arabic characters as modeling units. The transcriptions for the word image were modified to character level for both databases. We ended up with a total of 46 models in our recognition system. Table 7 shows the handwriting recognition accuracy rates for standard train-test configurations without using any augmented data, and Table 8 shows the handwriting recognition accuracy rates when using augmented data generated by standard techniques. In all experiments, the accuracy of handwriting recognition was evaluated by using the BLSTM network combined with the CTC layer.

TABLE 7. Accuracy Rates of the Systems using characters as models (BLSTM) – (NO Augmented data).

Database	Train-test configuration	BLSTM-CTC (Token Passing)		BLSTM-CTC (Word Beam Search)	
		CAR %	WAR %	CAR %	WAR %
IFN/ENIT	<i>abc-d</i>	85.33%	81.15%	86.70%	83.90%
	<i>bcd-a</i>	84.91%	79.77%	82.78%	81.19%
	<i>abcd-e</i>	80.13%	77.11%	81.84%	80.72%
	<i>abcde-f</i>	74.31%	72.61%	78.10%	77.20%
AHDB	70% training 20% testing 10% validation	80.92%	79.71%	82.74%	81.24%

As we mentioned earlier, it is not the best choice to use characters as models because Arabic characters have different position-dependent shapes. The accuracy rates of

TABLE 8. Accuracy Rates of the Systems using characters as models (BLSTM) – (using standard data augmentation).

	Train-test configuration	BLSTM-CTC (Token Passing)		BLSTM-CTC-WBS (Word Beam Search)	
		CAR %	WAR %	CAR %	WAR %
IFN/ENIT	<i>abc-d</i>	91.77%	89.25%	93.10%	91.19%
	<i>bcd-a</i>	89.85%	86.17%	91.41%	89.47%
	<i>abcd-e</i>	85.93%	83.91%	88.16%	86.15%
	<i>abcde-f</i>	81.74%	79.74%	84.20%	82.55%
AHDB	70% training 20% testing 10% validation	85.66%	84.42%	87.74%	86.52%

using characters as models are worse than the one obtained using character shapes as models. This because many characters have a similar shape, e.g., *س،ش،ص،ض*, which makes the task of the system in determining the real character more difficult. Besides, the shapes of the characters change according to their position in the word, which causes more confusion for the network for determining the correct class.

H. RESULTS OF USING ADAPTIVE DATA AUGMENTATION TECHNIQUE

In this section, we shall present the results of our adaptive data augmentation algorithm. First, let us look at the statistics of the IFN/ENIT database. Table 9 shows the number of occurrences of each character in the database along with its probability p_n . By looking at the table, we can notice that there is a huge difference between the appearance of some characters in the database. For example, the letter Alif (ا) is repeated more than 36 thousand times, compared to the letter Za (ظ), which we can barely see in the database. By knowing the characters sequence of each word in the database, we can calculate W_i , e.g., the word (الشرايع) has seven characters with each character has a certain p_n so:

$$W_{الشرايع} = \frac{\frac{1}{P_{Alif}} + \frac{1}{P_{Laam}} + \frac{1}{P_{Shin}} + \frac{1}{P_{Raa}} + \frac{1}{P_{Alif}} + \frac{1}{P_{Yaa}} + \frac{1}{P_{Ain}}}{7}$$

$$W_{الشرايع} = \frac{\frac{1}{0.17} + \frac{1}{0.092} + \frac{1}{0.0156} + \frac{1}{0.074} + \frac{1}{0.17} + \frac{1}{0.083} + \frac{1}{0.023}}{7}$$

$$= 22.253$$

Suppose that we want to augment 100,000 new samples to the original database. We need to find how many samples should we augment e.g. the word “الشرايع” out of the 100,000 so:

$$\text{Total No. of augmented samples for الشرايع} = [W_{الشرايع, normalized} \times 100,000]$$

$$= \frac{W_{الشرايع}}{\sum W_i (\text{sum of all classes wieghts in the database})} \times 100,000 = 140 \text{ samples.}$$

where, $\sum W_i = 15900$.

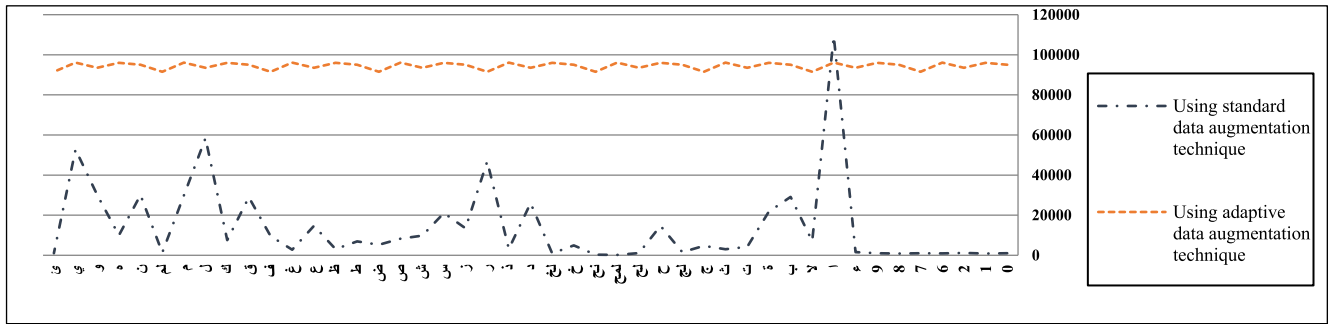


FIGURE 19. Using adaptive data augmentation vs. standard data augmentation on the IFN/ENIT database.

TABLE 9. IFN/ENIT database [21] statistics.

Character	No. of Occurrences	p_n	Character	No. of Occurrences	p_n
0	342	0.162	د	8657	4.106
1	279	0.132	ذ	1056	0.500
2	384	0.182	ر	15621	7.410
6	311	0.147	ز	4482	2.126
7	354	0.167	س	7012	3.326
8	284	0.134	ش	3270	1.551
9	341	0.161	ص	2764	1.311
ء	520	0.246	ض	1752	0.831
ا	36394	17.265	ط	2310	1.095
لا	2357	1.118	ظ	1029	0.488
ب	9718	4.610	ع	4902	2.325
ة	7259	3.443	غ	926	0.439
ت	1402	0.665	ف	3180	1.508
ث	1018	0.482	ق	9660	4.582
ج	1564	0.741	ك	2536	1.203
لج	539	0.255	ل	19475	9.238
ح	4896	2.322	م	9938	4.714
نج	365	0.173	ن	458	0.217
لنج	64	0.030	ن	10021	4.753
نح	100	0.047	ه	3318	1.574
خ	1640	0.778	و	10040	4.762
لخ	310	0.147	ي	17591	8.345

So, out of the new 100,000 augmented samples, 140 augmented samples will correspond to the word “الشرايع”. For all the words in the lexicon, this is repeated in order to find the number of augmented samples for each class. This will finally balance the database. We can choose any data augmentation technique to generate the new augmented data e.g. here we used a combination of standard data augmentation techniques (rotation and shear) to generate new 140 samples. Fig.19 shows the frequency distribution of the IFN/ENIT characters after using our adaptive data augmentation algorithm compared to using the standard data augmentation technique. We can see from the figure that if we used standard data-augmentation techniques the distribution of the data will remain highly skewed but by using our adaptive data-augmentation technique we can see that the distribution is approximately balanced now, e.g. infrequent letters like “Zaa ظ” and “Taa ت” have now a higher number of samples compared to using standard data augmentation technique.

Now let’s repeat these steps, but this time on the AHDB database. Assuming we are using characters as modeling units, then Table 10 shows the number of occurrences of each character in the database along with its p_n .

TABLE 10. AHDB [48] statistics.

Character	No. of Occurrences	p_n	Character	No. of Occurrence	p_n
ا	4400	0.141	ف	500	0.016
ب	1200	0.039	ق	400	0.013
ت	1300	0.042	ك	400	0.013
ث	1300	0.042	ل	1300	0.042
ح	300	0.01	م	2800	0.090
خ	700	0.022	ن	3100	0.1
د	300	0.01	ه	400	0.013
ذ	300	0.01	و	1300	0.042
ر	1400	0.045	ي	1500	0.048
س	2500	0.080	لا	600	0.019
ش	400	0.013	ة	2200	0.071
ظ	200	0.006	ى	300	0.01
ع	1300	0.042			

Fig.20 shows the frequency distribution of the characters after using our adaptive data augmentation algorithm compared to the standard data augmentation technique on the AHDB database.

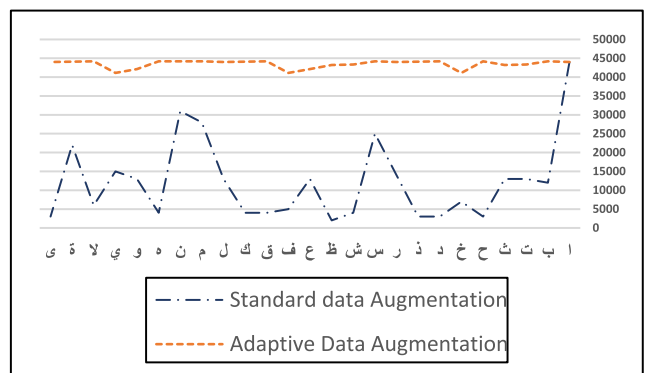


FIGURE 20. Using adaptive data augmentation vs. standard data augmentation on the AHDB database.

Table 11 shows the results of AlexNet and RNN’s holistic recognition on the IFN/ENIT and AHDB databases. The system was trained using the same configuration described

TABLE 11. AlexNet vs. RNN (Holistic Recognition) (using adaptive data augmentation).

Database	Train-test configurations	ALEXNET	RNN (LSTM)
		WAR %	WAR %
IFN/ENIT	<i>abc-d</i>	97.16%	96.91%
	<i>bcd-a</i>	95.85%	95.05%
	<i>abcd-e</i>	92.47%	91.87%
	<i>abcde-f</i>	91.77%	88.46%
AHDB	70% training 20% testing 10% validation	95.07%	93.87%

in Section 7.4 to evaluate their performances similarly. Both of the systems were trained this time by using our data augmentation algorithm described in Section.6.

Table 11 shows that recognition rates are significantly improved. The system performs better in all training test configurations with balanced data obtained by our algorithm.

Now, let us consider using the balanced data resampled using our algorithm in Section 6 as training data for the BLSTM network and by applying the same preprocessing and configurations steps used in Sections 7.3 and 7.4. Table 12 shows the results of handwriting recognition accuracy rates on the IFN/ENIT and AHDB databases augmented with our data augmentation algorithm. The results show that our data augmentation technique improves CAR% and the WAR% in all train-test configurations.

TABLE 12. Accuracy Rates of the Systems using characters as models (BLSTM) (using Adaptive data augmentation).

	Train-test configuration	BLSTM-CTC-WBS.	
		CAR %	WAR %
IFN/ENIT	<i>abc-d</i>	95.19%	96.19%
	<i>bcd-a</i>	93.33%	93.11%
	<i>abcd-e</i>	91.21%	91.96%
	<i>abcde-f</i>	90.90%	89.15%
AHDB	70% training 20% testing 10% validation	89.67%	88.41%

The same experiment was repeated, but this time we have used the character shapes as modeling units. Table 13 shows the accuracy rates of the system on all possible train-test configurations.

It can be seen from the table that the results are significantly better as compared to the results from the previous approaches. The system performs better with the balanced data resampled by our algorithm in all train-test configurations.

I. COMPARISON WITH OTHER STATE-OF-THE-ART SYSTEMS

In this section, we compare the results of recent state-of-the-art systems evaluated on the IFN/ENIT and AHDB databases with the results obtained by our systems. Table 14 presents a comparison of the results from the state-of-the-art

TABLE 13. Accuracy Rates of the Systems using character shapes as models (BLSTM) (using Adaptive data augmentation).

Database	Train-test configuration	BLSTM-CTC-WBS	
		CAR %	WAR %
IFN/ENIT	<i>abc-d</i>	99.73%	98.99%
	<i>bcd-a</i>	97.66%	96.91%
	<i>abcd-e</i>	96.23%	95.05%
	<i>abcde-f</i>	94.13%	93.57%
AHDB	70% training 20% testing 10% validation	98.49%	98.10%

TABLE 14. Comparison with other state-of-the-art systems evaluated on the IFN/ENIT and AHDB databases.

Author	WAR%		
	IFN/ENIT Database		
	Train-Test Configuration		
	<i>abc-d</i>	<i>abcd-e</i>	<i>abcde-f</i>
Al-Hajj, et al. [18]	90.96%	-	-
Dreuw et al. [19]	94.18%	88.78%	-
Graves et al. [17]	-	-	93.37%
Kessentini et al. [20]	-	79.69%	-
Pechwitz et al. [21]	91.80%	-	-
Rothacker et al. [22]	-	92.09%	-
Azeem et al. [23]	97.70%	93.44%	93.10%
Abandah et al. [24]	98.96%	93.46%	92.46%
Hamdani et al. [25]	-	-	92.20%
Elleuch et al. [26]	83.70%	-	-
Stahlberg et al. [27]	97.60%	93.09%	-
Ahmad et al. [28]	96.07%	94.93%	92.30%
Rabi et al. [29]	87.93%	-	-
Almodfer et al. [30]	92.55%	-	-
Ahmad et al. [31]	97.71%	94.76%	93.32%
Almodfer et al. [35]	-	-	91.50%
Present work BLSTM-CTC-WBS. (Using Adaptive Data Augmentation Algorithm)	98.99%	95.05%	93.57%

	WAR%
	AHDB Database
S. Al-Ma'adeed [48]	45.00%
S. Al-Ma'adeed et al. [54]	63.00%
Al-Nuzaili et al. [55]	89.29%
Hassan et al. [56]	99.08%
Hassan et al. [57]	96.32%
Present work BLSTM-CTC-WBS. (Using Adaptive Data Augmentation Algorithm)	98.10%

systems evaluated on the IFN/ENIT and AHDB databases. Many works were carried out on handwritten Arabic text recognition using IFN / ENIT and AHDB database. The best-reported systems in the literature on the evaluation sets (*e* and *f*) of the IFN/ENIT database were Ahmed and Fink [31]

and Graves *et al.* [17]. On the evaluation of set (d) Abandah *et al.* [24] were the best-reported system. For the AHDB the best-reported work in the literature was Hassan *et al.* [56]. The overall results of our systems are among the best in comparison with the other state-of-the-art system. It's also clear that our system integrated with the adaptive data augmentation technique has the highest accuracy rate on the IFN/ENIT database. For the AHDB database, the highest accuracy obtained was Hassan and Kadhm [57]. The authors proposed a handwritten Arabic recognition system that uses a bag of features with Support Vector Machine (SVM). Their system was evaluated using the AHDB database with 2072 images for the training set and 868 for the testing set while our system was implemented by using 7057 images for training and 2017 images for testing which explains why they obtained a higher accuracy rate than our system. Also, the other reason for obtaining higher accuracy than us is that SVM typically performs better on small datasets [45]. Another observation on the IFN/ENIT we can see the decrease in performance on set *f*. One of the reasons being that it was collected in a different environment and was added later to the dataset and is regarded as a relatively difficult set.

VIII. CONCLUSION

Offline recognition of handwritten Arabic texts is still a challenging area of research in the field of pattern recognition. Deep Learning Neural Networks (DLNN) has gained a reputation for taking care of numerous computer vision issues. Its application to the field of handwriting recognition has been shown to give fundamentally excellent results over conventional techniques. In this work, we have performed extensive experiments using different deep learning architectures to recognize handwritten Arabic text. Specifically, we used CNN, RNN, LSTM, Bi-LSTM, and CTC. Each system has been evaluated by using the IFN/ENIT and AHDB databases. Besides, we have proposed an adaptive data-augmentation algorithm that also attempts to deal with the issue of imbalanced classes. It has been shown that this method can solve rather effectively the data imbalance problem. Our proposed method is performed in such a way that the minority classes will be expanded by calculating the average probability of each class. The results show that the presented algorithm, when applied to the IFN/ENIT and AHDB databases can balance the distribution of the minority classes. The presented system in this paper is among the best state-of-the-art systems.

While all systems have produced excellent results, the accuracy rates were outperformed by using a CTC layer with a WBS decoder. During the tests, we have noticed that AlexNet and LSTM tend to over-fit and do not generalize well to the test data. Besides, the BLSTM system is robust to over-fit and has higher accuracy, especially to previously unseen data. During the training stage, experiments showed that augmenting new samples increases accuracy rates and prevent over-fitting, at the expense of an increase in the

training time of the network. Besides, using image transformations, e.g., vertical alignment of characters and removal of unnecessary white-space at the top and bottom of word images, could be suggested as some extra preprocessing steps.

As future work, we plan to use more advanced methods to enhance the current work, e.g., recursive recurrent nets with attention modeling [58], multi-dimensional LSTM network [59] and using other modeling options such as using sub-character and sub-core-shape as modeling units [31], which contributes to significant reduction in the number of models to be trained for the task of text recognition.

ACKNOWLEDGMENT

This research was supported by King Fahd University of Petroleum & Minerals (KFUPM).

REFERENCES

- [1] T. C. Wei, U. U. Sheikh, and A. A.-H.-A. Rahman, "Improved optical character recognition with deep neural network," in *Proc. IEEE 14th Int. Colloq. Signal Process. Its Appl. (CSPA)*, Mar. 2018, pp. 245–249.
- [2] M. H. H. Nashif, M. B. A. Miah, A. Habib, A. C. Moulik, M. S. Islam, M. Zakareya, A. Ullah, M. A. Rahman, and M. A. Hasan, "Handwritten numeric and alphabetic character recognition and signature verification using neural network," *J. Inf. Secur.*, vol. 09, no. 03, pp. 209–224, 2018.
- [3] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "ASTER: An attentional scene text recognizer with flexible rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2035–2048, Sep. 2019.
- [4] *Ethnologies: Languages of the World*, 22nd ed., SIL Int., Dallas, TX, USA, Feb. 2019.
- [5] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten arabic character recognition," *Cognit. Syst. Res.*, vol. 50, pp. 180–195, Aug. 2018.
- [6] A. A. A. Ali and M. Suresha, "An efficient character segmentation algorithm for recognition of arabic handwritten script," in *Proc. Int. Conf. Data Sci. Commun. (IconDSC)*, Mar. 2019, pp. 1–6.
- [7] M. Suresha and A. A. A. Ali, "Segmentation of handwritten text lines with touching of line," *Int. J. Comput. Eng. Appl.*, vol. 12, no. 6, pp. 1–12, 2018.
- [8] M. Al-Ayyoub, A. Nuseir, K. Alsmearat, Y. Jararweh, and B. Gupta, "Deep learning for arabic NLP: A survey," *J. Comput. Sci.*, vol. 26, pp. 522–531, May 2018.
- [9] J. L. Vásquez, A. G. Ravelo-García, J. B. Alonso, M. K. Dutta, and C. M. Travieso, "Writer identification approach by holistic graphometric features using off-line handwritten words," *Neural Comput. Appl.*, vol. 2018, pp. 1–14, Mar. 2018.
- [10] F. Nashwan, M. Rashwan, H. Al-Barhamtoshy, S. Abdou, and A. Moussa, "A holistic technique for an arabic OCR system," *J. Imag.*, vol. 4, no. 1, p. 6, Dec. 2017, doi: 10.3390/jimaging4010006.
- [11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*. New York, NY, USA: ACM, 2006, pp. 369–376.
- [12] R. Maalej and M. Kherallah, "Convolutional neural network and BLSTM for offline arabic handwriting recognition," in *Proc. Int. Arab Conf. Inf. Technol. (ACIT)*, Nov. 2018, pp. 1–6.
- [13] S. K. Jemni, Y. Kessentini, S. Kanoun, and J.-M. Ogier, "Offline arabic handwriting recognition using BLSTMs combination," in *Proc. 13th IAPR Int. Workshop Document Anal. Syst. (DAS)*, Apr. 2018, pp. 31–36.
- [14] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández, "Unconstrained on-line handwriting recognition with recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 577–584.
- [15] D. Yogatama, C. Dyer, W. Ling, and P. Blunsom, "Generative and discriminative text classification with recurrent neural networks," 2017, *arXiv:1703.01898*. [Online]. Available: <http://arxiv.org/abs/1703.01898>
- [16] I. Ahmad and S. A. Mahmoud, "Arabic bank check processing: State of the art," *J. Comput. Sci. Technol.*, vol. 28, no. 2, pp. 285–299, Mar. 2013.
- [17] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 545–552.

- [18] R. Al-Hajj, R. Al-Hajj, C. Mokbel, C. Mokbel, L. Likforman-Sulem, and L. Likforman-Sulem, "Combination of HMM-based classifiers for the recognition of arabic handwritten words," in *Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 2, Sep. 2007, pp. 959–963.
- [19] P. Dreuw, S. Jonas, and H. Ney, "White-space models for offline arabic handwriting recognition," in *Proc. 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
- [20] Y. Kessentini, T. Paquet, and A. Ben Hamadou, "Off-line handwritten word recognition using multi-stream hidden Markov models," *Pattern Recognit. Lett.*, vol. 31, no. 1, pp. 60–70, Jan. 2010.
- [21] M. Pechwitz, H. El Abed, and V. Märgner, "Handwritten Arabic word recognition using the IFN/ENIT-database," in *Guide to OCR for Arabic Scripts*. London, U.K.: Springer, 2012, pp. 169–213.
- [22] L. Rothacker, S. Vajda, and G. A. Fink, "Bag-of-features representations for offline handwriting recognition applied to Arabic script," in *Proc. Int. Conf. Frontiers Handwriting Recognit.*, Sep. 2012, pp. 149–154.
- [23] S. A. Azeem and H. Ahmed, "Effective technique for the recognition of offline arabic handwritten words using hidden Markov models," *Int. J. Document Anal. Recognit. (IJ DAR)*, vol. 16, no. 4, pp. 399–412, Dec. 2013.
- [24] G. A. Abandah, F. T. Jamour, and E. A. Qaralleh, "Recognizing handwritten arabic words using grapheme segmentation and recurrent neural networks," *Int. J. Document Anal. Recognit.*, vol. 17, no. 3, pp. 275–291, Sep. 2014.
- [25] M. Hamdani, P. Doetsch, M. Kozielski, A. E.-D. Mousa, and H. Ney, "The RWTH large vocabulary arabic handwriting recognition system," in *Proc. 11th IAPR Int. Workshop Document Anal. Syst.*, Apr. 2014, pp. 111–115.
- [26] M. Elleuch, N. Tagougui, and M. Kherallah, "Deep learning for feature extraction of arabic handwritten script," in *Proc. Int. Conf. Comput. Anal. Images Patterns*. Cham, Switzerland: Springer, 2015, pp. 371–382.
- [27] F. Stahlberg and S. Vogel, "The QCRI recognition system for handwritten arabic," in *Proc. Int. Conf. Image Anal. Process.*, Cham, Switzerland: Springer, 2015, pp. 276–286.
- [28] I. Ahmad and G. A. Fink, "Multi-stage HMM based arabic text recognition with rescoring," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2015, doi: 10.1109/ICDAR.2015.7333862.
- [29] M. Rabi, M. Amrouch, Z. Mahani, and D. Mammass, "Recognition of cursive Arabic handwritten text using embedded training based on HMMs," in *Proc. Int. Conf. Eng. MIS (ICEMIS)*, Agadir, Morocco, Sep. 2016, pp. 1–5.
- [30] R. Almodfer, S. Xiong, M. Mudhsh, and P. Duan, "Enhancing AlexNet for arabic handwritten words recognition using incremental dropout," in *Proc. IEEE 29th Int. Conf. Tools Artif. Intell. (ICTAI)*, Boston, MA, USA, Nov. 2017, pp. 663–669.
- [31] I. Ahmad and G. A. Fink, "Handwritten arabic text recognition using multi-stage sub-core-shape HMMs," *Int. J. Document Anal. Recognit. (IJ DAR)*, vol. 22, no. 3, pp. 329–349, Sep. 2019.
- [32] A. Al-Jallad, "Stefan Wening in collaboration with Geoffrey Khan, Michael P. Streck and Janet C.E. Watson (eds), *the semitic languages: An international handbook*," *J. Semitic Stud.*, vol. 58, no. 2, pp. 395–398, 2013.
- [33] A. Rehman, D. Mohamad, and G. Sulong, "Implicit vs explicit based script segmentation and recognition: A performance comparison on benchmark database," *Int. J. Open Problems Compt. Math.*, vol. 2, no. 3, pp. 352–364, 2009.
- [34] A. Choudhary, "A review of various character segmentation techniques for cursive handwritten words recognition," *Int. J. Inf. Comput. Technol.*, vol. 4, no. 6, pp. 559–564, 2014.
- [35] R. Almodfer, S. Xiong, M. Mudhsh, and P. Duan, "Multi-column deep neural network for Offline arabic handwriting recognition," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2017, pp. 260–267.
- [36] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
- [37] A. Ashiqzaman and A. K. Tushar, "Handwritten arabic numeral recognition using deep learning neural networks," in *Proc. IEEE Int. Conf. Imag., Vis. Pattern Recognit. (icVPR)*, 2017, pp. 1–4.
- [38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
- [39] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, S. C. Kremer and J. F. Kolen, Eds. Piscataway, NJ, USA: IEEE Press, 2001.
- [40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 8, no. 9, pp. 1735–1780, 1997.
- [41] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2017, *arXiv:1801.01078*. [Online]. Available: <http://arxiv.org/abs/1801.01078>
- [42] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 1017–1024.
- [43] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.
- [44] H. Scheidl, S. Fiel, and R. Sablatnig, "Word beam search: A connectionist temporal classification decoding algorithm," in *Proc. 16th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Aug. 2018, pp. 253–258.
- [45] L. Marceau, L. Qiu, N. Vandewiele, and E. Charton, "A comparison of deep learning performances with other machine learning algorithms on credit scoring unbalanced data," 2019, *arXiv:1907.12363*. [Online]. Available: <http://arxiv.org/abs/1907.12363>
- [46] Wikipedia. (2020). *Letter Frequency*. Accessed: Apr. 6, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Letter_frequency
- [47] Intellaren. (2020). *A Study of Arabic Letter Frequency Analysis*. Accessed: Mar. 9, 2020. [Online]. Available: <http://www.intellaren.com/articles/en-a-study-of-arabic-letter-frequency-analysis>
- [48] S. Al-Ma'adeed, D. Elliman, and C. A. Higgins, "A data base for Arabic handwritten text recognition research," in *Proc. 8th Int. Workshop Frontiers Handwriting Recognit.*, Aug. 2002, pp. 485–489.
- [49] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Noguees, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1285–1298, May 2016.
- [50] A. Graves. (2016). *RNNLIB: A Recurrent Neural Network Library for Sequence Learning Problems*. [Online]. Available: <http://sourceforge.net/projects/rnnl>
- [51] *MATLAB Version 2019A*, MathWorks Inc., Natick, MA, USA, 2019.
- [52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, and M. Devin, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
- [53] G. Bradski and A. Kaehler, "OpenCV," *Dr. Dobbs's J. Softw. Tools*, vol. 3, pp. 120–125, 2000.
- [54] S. Alma'adeed, "Recognition of off-line handwritten arabic words using neural network," in *Proc. Geometric Model. Imaging–New Trends (GMAI)*, Jul. 2006, pp. 141–144.
- [55] Q. Al-Nuzaili, S. Al-Maadeed, H. Hassen, and A. Hamdi, "Arabic bank cheque words recognition using Gabor features," in *Proc. IEEE 2nd Int. Workshop Arabic Derived Script Anal. Recognit. (ASAR)*, Mar. 2018, pp. 84–89.
- [56] A. K. A. Hassan, B. S. Mahdi, and A. A. Mohammed, "Arabic handwriting word recognition based on scale invariant feature transform and support vector machine," *Iraqi J. Sci.*, vol. 60, pp. 381–387, Feb. 2019.
- [57] A. K. A. Hassan and M. S. Kadhmi, "Arabic handwriting text recognition based on efficient segmentation, DCT and HOG features," *Int. J. Multimedia Ubiquitous Eng.*, vol. 11, no. 10, pp. 83–92, Oct. 2016.
- [58] C.-Y. Lee and S. Osindero, "Recursive recurrent nets with attention modeling for OCR in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2231–2239.
- [59] T. Bluche, J. Louradour, and R. Messina, "Scan, attend and read: End-to-end handwritten paragraph recognition with MDLSTM attention," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Nov. 2017, pp. 1050–1055.



MOHAMED ELTAY received the B.Eng. degree (Hons.) in electronic engineering from the University of Multimedia, Melaka, Malaysia, in 2013. He is currently pursuing the master's degree with the King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. His main areas of research interest are digital signal processing, machine learning, and pattern recognition.



ABDELMALEK ZIDOURI (Senior Member, IEEE) received the M.Sc. degree in control engineering from Bradford University, U.K., in 1984, and the D.Eng. in applied electronics from the Tokyo Institute of Technology, Japan, in 1995.

He is an Associate Professor with the Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. His research interests are in the field of signal processing and pattern recognition. In particular, character recognition and document image analysis. He has published many refereed journal articles and conference papers. He has supervised many projects, theses, and dissertations. He was the Head of the EE Department, Digital Signal Processing Group. He is currently a Seminar Coordinator of the Department. He is a member of the Engineering Education Society, the Signal Processing Society, and the Communications Society.



IRFAN AHMAD received the M.S. degree in computer science from the King Fahd University of Petroleum and Minerals (KFUPM) and the Ph.D. degree in computer science from TU Dortmund, Germany, in 2017. He is actively involved in research in the areas of pattern recognition and machine learning. He is an Assistant Professor with the Department of Information and Computer Science, KFUPM. He has worked on several research projects funded by KACST and KFUPM.

He has published several articles in high quality journals and international conferences. He has also published a book chapter and has three U.S. patents under his name. He regularly reviews articles for well-known journals and conferences in his area of research in addition to being a program committee member for some reputed international conferences.

• • •