Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 3, No. 3, December 2017.

ARABIC HANDWRITTEN CHARACTER RECOGNITION BASED ON DEEP CONVOLUTIONAL NEURAL NETWORKS

Khaled S. Younis[1]

(Received: 24-Jun.-2017, Revised: 06-Sep.-2017 and 21-Oct.-2017, Accepted: 31-Oct.-2017)

ABSTRACT

The automatic analysis and recognition of offline Arabic handwritten characters from images is an important problem in many applications. Even with the great progress of recent research in optical character recognition, a few problems still wait to be solved, especially for Arabic characters. The emergence of Deep Neural Networks promises a strong solution to some of these problems. We present a deep neural network for the handwritten Arabic character recognition problem that uses convolutional neural network (CNN) models with regularization parameters such as batch normalization to prevent overfitting. We applied the Deep CNN for the AIA9k and the AHCD databases and the classification accuracies for the two datasets were 94.8% and 97.6%, respectively. A study of the network performance on the EMNIST and a form-based AHCD dataset were performed to aid in the analysis.

KEYWORDS

Convolutional neural network, Deep learning, Optical character recognition, Arabic handwritten character recognition, EMNIST.

1. INTRODUCTION

The field of optical character recognition (OCR) is very important, especially for offline handwritten recognition systems. Offline handwritten recognition systems are different from online handwritten recognition systems [1]. The ability to deal with large amounts of script data in certain contexts will be invaluable. One example of these applications is the automation of the text transcription process applied on ancient documents considering the complex and irregular nature of writing [2]. Arabic optical text recognition is experiencing slow development compared to other languages [3].

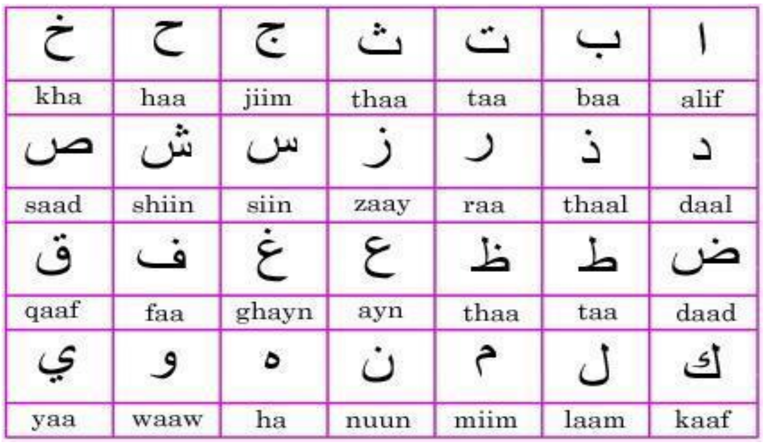
One problem with recognizing the Arabic alphabet is that many characters have similar shapes but with varying locations of dots relative to the main part of the character. Figure 1 shows the isolated alphabet of the Arabic language. As can be seen at the top row, the three characters on the left have a similar main part but the dot on the “Kha” is above while, for the “Jiim”, the dot is below the main part and the “Haa” has no dots at all. It is noteworthy that handwritten characters are more challenging, since human writers tend to combine dots and use dashes instead or change the shape of characters as can be seen in Figure 2, which shows 48 handwritten samples of the same letter "Ayn" that were used in previous work [4].

Moreover, the Arabic alphabet is widely used by many people from different countries including all Arab countries in addition to being used in the Persian, Urdu and Pashto languages. It would be great to use Arabic handwritten character recognition (AHCR) to convert many documents into digital format that can be accessed electronically. Applications include: reading postal addresses off envelops and automatically sorting mail, helping the blind to read, reading customer-filled forms (government forms, insurance claims, application forms), automating offices, archiving and retrieving text and improving human-computer interfaces.

Deep Learning (DL) is a new application of machine learning for learning representation of data. DL algorithms have taken the top place in the object recognition field due to the great performance improvement they have provided [5], [30].

This paper is an extended version of a short paper that was presented at the international conference "New Trends in Information Technology (NTIT) 2017", 25-27 April 2017, Amman. Jordan.

1. Khaled S. Younis is with the Department of Computer Engineering, The University of Jordan, Amman, Jordan. Email: `Younis@ju.edu.jo`.



خ	ح	ج	ث	ت	ب	ا
kha	haa	jiim	thaa	taa	baa	alif
ص	ش	س	ز	ر	ذ	د
saad	shiin	siin	zaay	raa	thaal	daal
ق	ف	غ	ع	ظ	ط	ض
qaaf	faa	ghayn	ayn	thaa	taa	daad
ي	و	ه	ن	م	ل	ك
yaa	waaw	ha	nuun	miim	laam	kaaf

Figure 1. The Arabic alphabet.

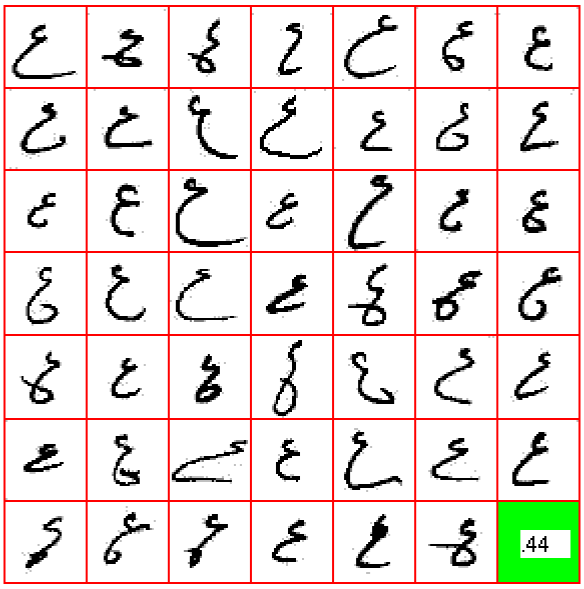


Figure 2. Sample of 48 handwritten "Ayn" letters.

Deep Learning (DL) is a new application of machine learning for learning representation of data. DL algorithms have taken the top place in the object recognition field due to the great performance improvement they have provided [5], [30].

Convolutional Neural Networks (CNNs) are a type of neural networks that are applied in many fields and provide efficient solutions in many problems, where there is some translation invariance like some applications of object recognition and speech recognition. However, CNN DL solutions require a lot of training samples, which places computational requirements on the system. Nevertheless, the accelerating progress and availability of low-cost computer hardware, high-speed networks and software for high-performance distributed computing encouraged the use of computationally expensive techniques. For example, Cecotti [6] used Graphical Processing Units (GPUs) and High-Performance Clusters (HPCs) to classify isolated characters of 9 databases using computationally expensive techniques.

There are several frameworks for Deep Learning. One of the most popular libraries is TensorFlow, that was released by Google in 2015 [7]. It is an open source code written in C++ programming language and capable of using GPUs very well. Another simpler framework is Keras [8], which is a higher-level API built on top of TensorFlow. Keras uses Python for programming, which makes writing programs easier than native TensorFlow codes.

Therefore, in this paper, we will discuss the building of a robust CNN DL model for solving the problem of AHCR using TensorFlow/Keras. This model is expected to outperform traditional AHCR algorithms that depend on feature extraction and classification and can be applied on huge and different databases efficiently without the need for feature engineering and extremely long training time.

The contributions of this research are: (i) Reviewing state-of-the-art research in AHCR (ii) Proposing robust architecture for CNN DL for solving the AHCR problem (iii) Studying the effect of different regularization techniques and network parameters on the performance on very large size AHCR databases (iv) Utilizing the functionalities offered by TensorFlow and Keras libraries for AHCR and (v) Achieving the highest accuracy on the AHCR problem.

The rest of the paper is organized as follows; Section 2 discusses related work in the field of AHCR. Section 3 describes the motivation for the proposed solution as well as the different components of its architecture, Section 4 introduces the experiments performed in a scientific way, including the results obtained and Section 5 discusses conclusions derived from the results and presents plans for future work.

2. RELATED WORK

Algorithms designed to recognize handwritten characters are still less successful than those for printed characters, mainly due to the diversity in handwritten character shapes and forms. Arabic character recognition is an important problem, since it is a step that may be needed in the more challenging Arabic word or sentence recognition problem [9]. Character segmentation to separate the word into characters is another challenging problem. The character recognition problem is related to the simpler problem of Arabic numeral recognition which has recently attained great results [10].

Various methods have been proposed and high recognition rates are reported for the handwritten English and Chinese characters. However, in this section, we are going to present only the most competitive related work solving the AHCR problem.

Many algorithms in the past concentrated on finding structural features (such as the presence of loops, the orientation of curves, …etc.) or statistical features (such as moments, histogram of gray level distribution, …etc.) [11]. These features try to maximize the interclass variability while minimizing the intra-class variability and were fed to a classifier.

Some algorithms are considered segmentation-based recognition systems; this means that their experimental results depend on segmenting the words before recognizing the characters. The IFN/ENIT database was used in Al-abodi and Li [12], who had proposed a recognition system based on geometrical features of Arabic characters. The average recognition accuracy is 93.3%. Other works that used IFN/ENIT for segmentation-based character recognition such as [33] also achieved similar performance using three main modules: preprocessing, feature extraction and recognition. However, we will not discuss such system in this paper. Even though the IFN/ENIT database [28] is available, it is designed for classifying words and letter segmentation is required before character recognition is performed. In addition, it is considered small and does not contain enough representative samples. Therefore, it was deemed unsuitable for CNN DL architecture evaluation.

Arabic characters have different forms depending on the location of the characters in the word. Hidden Markov Models (HMM) assume each letter is a state and using the context leads to better classification of Arabic handwritten word as in [37]. Nevertheless, recent work [39] discusses the limitations of HMM in terms of the need for manual extraction of features, which requires prior knowledge of the language and is robust to handwriting diversity and complexity. CNN applications to direct word recognition have been discussed in [39] and [41]. Use of Bidirectional Long Short Term Memory (BLSTM) networks is proved useful in other languages, but the application to Arabic language is of great interest. However, the lack of very large datasets and the layout of Arabic text are causing problems in implementation. One can also use character segmentation followed by recognition. For the latter, they used LSTM [38] with convolution to construct bounding boxes for each character. We then pass the segmented characters to a CNN for classification and then reconstruct each word according to the results of classification and segmentation. It was shown that character segmentation had given better performance confirming the intuition that the much smaller scope of the model's initial feature representation problem for characters as opposed to words and final labeling problem helped boost the performance. There is great diversity in handwriting for particular words/characters among writers, thus making the task of recognizing all of the different ways in which a character or word is written very challenging. An important aspect is the availability of huge dataset to train the deep network. Since there is no such database for Arabic words, the analysis of isolated character recognition is important and may be included in the system for segmentation-based word or sentence recognition.

In 2014, Torki et al. [13] built their own database of about nine thousand characters. They called it the AlexU Isolated Alphabet (AIA9k) database. Then, they extracted three window-based gradient-based descriptors: Histogram of Oriented Gradients (HOG) [14], Speeded-Up Robust Features (SURF) [15] and Scale Invariant Feature Transform (SIFT) [16]. In addition, they extracted two texture-based descriptors and tried 4 classifiers (Logistic regression, ANN, SVM-Linear and SVM-RBF) on their database. The best achieved accuracy was 94.28% using SVM-RBF on SIFT features. The 75 characters that were misclassified are shown in Figure 3. While there are some characters that are not that difficult to classify, some characters are indeed confusing.

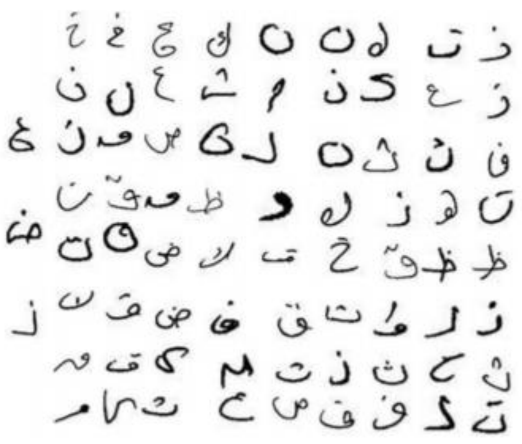


Figure 3. Misclassified characters from the AIA9K dataset using the method of [13].

In 2015, Lawgali [11] published a survey about Arabic Character Recognition and none of the algorithms mentioned used deep learning. However, also in 2015, Elleuch [9] introduced an Arabic handwritten character recognition using Deep Belief Neural Networks. It does not require any feature engineering. The input is simply the raw data or grayscale pixel values of the images. The approach was tested on the HACDB database [17] that contains 6600 shapes of handwritten characters written by 50 persons. The dataset is divided into a training set of 5280 images and a test set of 1320 images. The result was promising on the character recognition task with 97.9% accuracy but discouraging on the word recognition database with an accuracy of less than 60%.

In 2017, Elleuch [18] continued working on the DBN and stack of feature extractors, such as Restricted Boltzmann Machine (RBM) and Auto-Encoder and reported the results on character recognition that was in fact similar (97.8%) to the previous work. These are very promising results and demonstrate the superiority of DL methods in AHCR.

Nevertheless, it must be mentioned that the HACDB database is considered an easy and clean database and there are well defined main parts of the different letter forms among 66 different classes. On the other hand, in character recognition, it is harder to classify similar letters which are only different by a dot. HACDB are much easier to classify and have three times classes as the AIA9k database.

In 2017, El-Sawi et al. [19] collected the Arabic Handwritten Character Dataset (AHCD) of 16800 images of isolated characters. They built a CNN Deep learning architecture to train and test the dataset. They used optimization methods to increase the performance of CNN. Their proposed CNN gave an average 94.9% classification accuracy on testing data.

We can see that a few techniques of deep learning have proven their usefulness for the AHCR problem and hence we will explain in the Motivation section next why we decided to propose the following architecture.

3. Proposed Architecture

In this section, we describe the criterion behind the decisions taken during the design phase and the architecture parameters for the models used for solving the AHCR problem.

3.1 Motivation

It is foreseen that, due to the success of modern neural network architectures, state-of-the-art handwritten recognition systems will either go towards hybrid systems (Deep Networks with some segmentation and feature extraction) or pure neural recognizers featuring deep architectures [20]. The discussed related work has demonstrated the inefficiency of selecting the right feature and going through the preprocessing stages. The goal of this paper is to study the CNN DL approach that particularly makes use of the Convolution layer to leverage three ideas that help improve the classification network: sparse interaction, parameter sharing and equivalent representation [21].

We will apply a robust CNN architecture to the Arabic characters AIA9k and AHCD databases as a case study with enough samples to validate the assumptions and give meaningful feedback. We will use CNN capability to extract features and train for recognition instead of extracting a large set of gradient or textural features as it was done in Torki et al. [13]. Moreover, we will use recent techniques of optimization and regularization, such as Batch Normalization [22] and Varying Learning Rate, to deal with training neural network issues. These were not used in the work of El-Sawy et al. [19], for example, as they used fixed learning rate and didn't normalize the mini-batches during training. Moreover, by making a large CNN network with many layers, it becomes more capable to detect more features automatically. Hence, using different numbers of convolutional layers with different numbers of filters should help us achieve better accuracy.

To solve the problem of latency in processing the data, GPUs are used as suggested by Ciresan et al. [23], who trained and tested the CNN network using a committee of classifiers and reduced the error rate of MNIST dataset [24] to 2.7%. For this reason, we decided to choose Keras and TensorFlow as the developing environment, since there is a GPU-enabled TensorFlow with support for CUDA acceleration.

3.2 CNN Architecture

Convolutional neural networks can convert the input structure through each layer of the network seamlessly to extract automatically the features of the images.

CNNs are based on a mathematical operation called convolution. A convolution is a multiplication operation of each pixel in the image with each value in the kernel, which is in turn another matrix and then summing the products. The key advantage of using the convolution operation is generating many images from the original image that enhance different features extracted from the original image, which leads to making the classification process more powerful [29].

In CNN, we use different types of layers as shall be explained shortly. First, the convolution layer, also called a feature extractor, extracts features from the input image. Initially, CNN does not know where exactly the features (shapes) in the image will be located; so, it tries to find them everywhere in the image by using a matrix called filter. Each filter represents a specific feature. CNN applies the convolution operation by a sliding filter in the image and multiplies each pixel in the image with each value in the filter. Then, this operation is repeated for other features (filters) and the output of this layer will be a set of filtered images [29].

In modern deep learning libraries, some consider a second layer called "Nonlinearity layer". In this layer, the Rectified Linear Unit (ReLU) activation function of the neurons is implemented to produce an output after each convolution [34]. ReLU is an element-wise operation (applied per pixel) to introduce non-linearity in our network. Since convolution is a linear operation, element-wise matrix multiplication and addition, so we add nonlinearity using ReLU. This operation converts each negative pixel in a feature map into zero and keeps each positive pixel.

Batch Normalization is a normalization and regularization technique proposed by Ioffe and Szegedy [22] to address the following issues that appear during the training process of deep neural networks:

1. Internal Covariate Shift: which refers to the change in the distribution of input of each layer

(features) that is affected by parameters in all input layers in which a small change in the network can significantly affect the entire network; and

2. Vanishing Gradient in saturating nonlinear functions: such as tanh and sigmoid, which are prone to get stuck in the saturation region as the network grows deeper despite the proposed solutions to carefully initialize the network, using small learning rate or replacing these functions by ReLU function.

Our system suggests the use of Batch Normalization as a part of the network architecture and it was experimentally proven to cause an improvement in terms of speed and accuracy. The Batch Normalization layer is added just before the nonlinearity and especially after the convolutional layers to limit its output away from the region of saturation using the mean and variance.

The Pooling or subsampling layer reduces the dimensionality of each filtered image, but preserves the most important features in the previous layer. Pooling can be of different types: Maximum, Average Sum, …etc. The output will have the same number of images, but they will each have fewer pixels. This is also helpful in managing the computational load [36]. The pooling operation is demonstrated in Figure 4. However, it is argued that max-pooling can be redundant and could be replaced by purely using convolutional layer with increased stride without loss in accuracy [35].

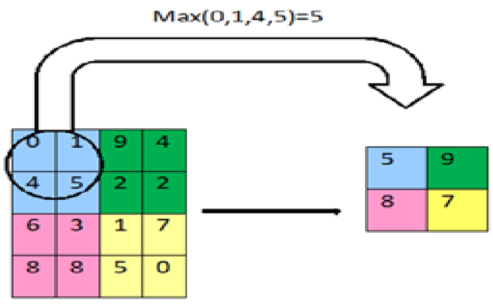


Figure 4. Pooling operation.

Dropout layers are also used in convolutional neural networks with the aim of reducing overfitting. This layer "drops out" a random set of neurons in that layer by setting their activation to zero. It makes sure that the network can generalize to test data by getting weights that are insensitive to training samples. Dropout is used during training with different percentages of total number of neurons in each layer [26].

Finally, the fully connected layers are the basic building blocks of traditional neural networks. They treat the input as one vector instead of two dimensional arrays. Full connection implies that every neuron in the previous layer is connected to every neuron in the next layer. The output from convolutional and pooling layers represents high level features and fully connected layers used to classify material (input images) into the appropriate class based on the training of the dataset [36].

Figure 5 shows the base architecture of the AHCR proposed network. Other modifications will be explained in the next sections.

As can be seen, the general network we designed has three convolutional layers followed by a fully connected layer as hidden layers. Max pooling can be ignored. The first layers are the input layers that take input of shape 28x28 or 32x32 pixels of grayscale characters depending on the size of input samples (database), then a convolutional layer of 24 filter map of size 6x6 and stride 1, followed by batch normalization, ReLU activation function and dropout of 1.0.

Dropout technique at the end of some layers is used with a "keep probability" parameter of 0.5. This means that at each training iteration, half of the neurons of the last layer get activated while the other half activation is set to zero. This tends to prevent our network from overfitting by not building a model so tightly tied to the training samples.

In the next layers, the same order of layers is used with increased number of convolutional filter map of 48 filters of size 5x5 and stride 2 and 64 filters of size 4x4 and stride 2, respectively.

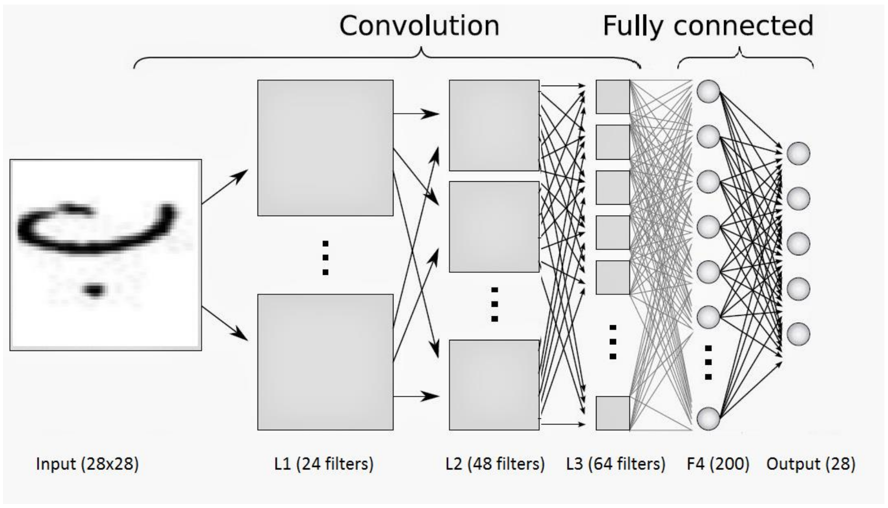


Figure 5. Proposed CNN architecture for the AHCR problem.

Finally, a fully connected layer of 200 neurons is used before the output layer of 28 neurons to match the number of classes (Arabic alphabet). We used the Softmax activation function to output probabilities between 0 and 1 for each class representing the confidence that a certain character belongs to a specific class.

For updating the weights during training, we used the Categorical Cross-Entropy [25] as a cost function which is the appropriate cost function for multi-class classification problems. We used Adam Optimizer to find the minima of the cost function with a varying learning rate [27] that recalculates its value after each batch.

4. EXPERIMENTS

4.1 Databases

In this section, we describe the different publicly available datasets that were used to evaluate the proposed network.

4.1.1 Arabic Handwritten Character Dataset (AHCD)

The dataset is composed of 16,800 characters written by 60 participants; the age range is from 19 to 40 years and 90% of participants are right-handed. Each participant wrote each character (from "Alef" to "Yeh") ten times. The forms were scanned at a resolution of 300 dpi. The database is partitioned into two sets: a training set (13,440 characters to 480 images per class) and a test set (3,360 characters to 120 images per class) [19].

4.1.2 AlexU Isolated Alphabet (AIA9K) Dataset

This dataset introduces a compact 9K novel dataset of 28 classes that represent isolated Arabic handwritten alphabet of 32x32 pixels [13]. AIA9K dataset was collected from 107 volunteer writers, between 18 and 25 years old, who are B.Sc. or M.Sc. students. The writers were 62 females and 45 males. Each writer wrote all of the Arabic letters 3 times. The total valid number of collected characters is 8,737 letters; this novel dataset can be requested from the authors of the paper mentioned in [13]. A sample of the dataset is shown in Figure 3. These are 75 characters that were misclassified in one of the experiments in [13].

4.2 Results

This section introduces the results obtained by the proposed AHCR network. Two subsections will describe the results of applying the network to classify AIA9K dataset and the AHCD datasets, subseq-uently. A subsection will compare the results obtained using the proposed approach with those of other approaches. Next, we will describe the results of applying the proposed network on Latin (English) characters. Finally, we will generate a derived database from the AHCD database, where only samples of each group of characters had the same shape (major stroke), then we will discuss the application of the proposed methodology on this dataset.

4.2.1 Results of the AHCR System Using The AHCD Dataset

The results we obtained after testing the proposed network described in sub-section 3.2 on the AHCD Arabic isolated alphabet dataset are described here. We divided the data into three parts; training, validation and testing, with ratios of 70%, 15% and 15% for each set, respectively. Then, we ran training for 10 epochs and 100 batch sizes. We obtained an accuracy of 92% on the test set at the end of the training.

We increased the number of epochs to 20 and 28, respectively. Notable improvements have been obtained and accuracy increased to 93% and 94.5%, respectively. In the next step, we increased the number of filters of the first convolutional layer from 24 to 72, the second convolutional layer from 48 to 144, the third convolutional layer from 64 to 192 and increased the number of the fully connected layer neurons from 200 to 400. Test accuracy improved to 94.7%.

Analysis of the difference in accuracies of training (100%), validation (97.5%) and testing (94.7%) revealed a gap that is an indication of overfitting. One way of improving generalization is to increase the size of training data. A simple shift of the input image to the left by 1 pixel will result in a total different input for the network, while it does not affect the actual class. Data augmentation techniques are a way to artificially expand the dataset. Some popular augmentation examples are horizontal flips, vertical flips, random crops, translations and rotations. Data augmentation was deemed necessary to improve the network performance. We increased the number of training images from ~13k to ~80k using translation in both horizontal and vertical directions by 3 pixels, rotation of +10 and -10 degrees and by adding Gaussian noise with zero mean and a standard deviation of 5. Horizontal or vertical flipping was deemed unsuitable for our application. However, we have seen significant performance gain merely by using translation which was reflected in obtaining a testing data accuracy of 96.7%. On the other hand, when we used all the 80k images and after training for 18 epochs, accuracy jumped to 97.6%.

Figure 6 shows that the training and validation accuracies changed during training for 18 epochs on 80k augmented dataset. It is obvious that the gap between the two metrics was insignificant after the initial epochs. The small jump on epoch 18 was an indication of the early stopping function that avoids overfitting and stop training.

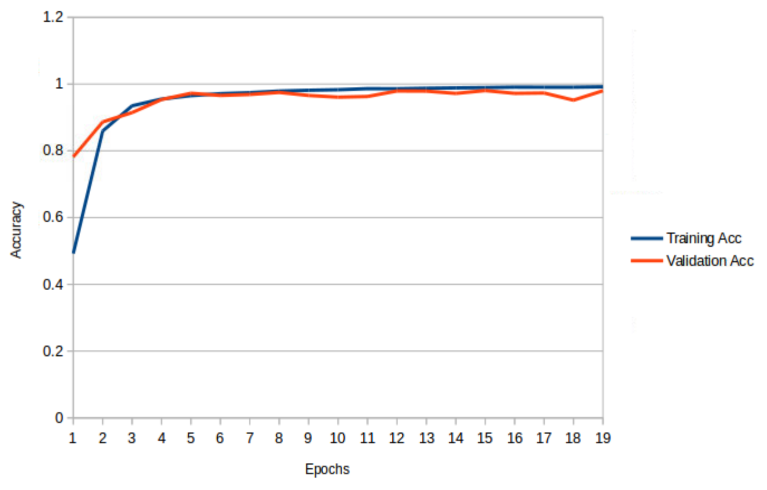


Figure 6. Training accuracy and validation accuracy during training.

Figure 7 shows the training and validation loss curves as functions of training epochs. It is worth mentioning that validation loss decreased significantly at epoch 4 and stayed very small until the end of epoch 19.

Using varying learning rate also helped reach lower minimum of the loss function. This technique is very essential in many optimization algorithms.

Before we used data augmentation to train the network, many experimental setups have been tested to improve the network performance [21] as summarized below:

- We tried to increase the network capacity by increasing the size of the network via adding another fully connected layer of 200 neurons. However, testing accuracy decreased.
- Test accuracy decreased to 93.5% when we reduced the number of neurons of the fully connected layer to 150 neurons.
- Test accuracy increased to 94.7% when we tried to double the number of filters to the three convolutional layers to 48, 96 and 128, respectively and increased the number of neurons of the fully connected layer from 200 to 300.
- No change in performance occurred when we tripled the number of convolutional layer filters and made the fully connected layer neurons 400, which indicates that the network size was adequate.
- Thresholding the grayscale images to convert them into binary images decreased the accuracy to 92.1%, which highlights the benefits of grayscale information.

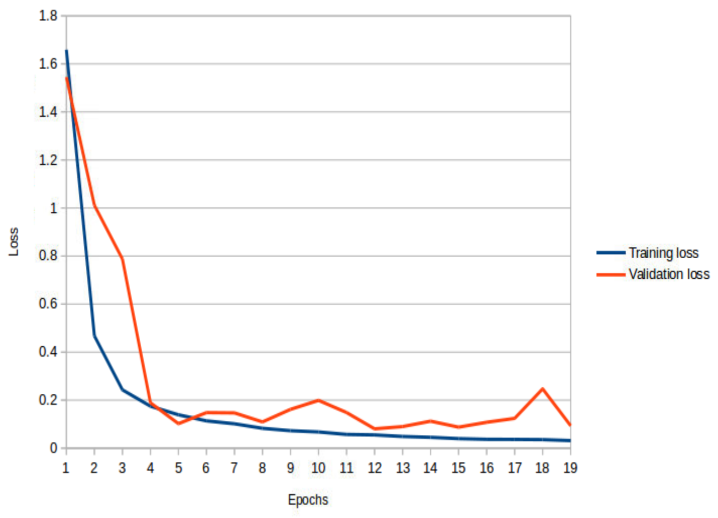


Figure 7. Training loss and validation loss.

In terms of the effect of changing the regularization parameters:

- Test accuracy decreased a lot when we trained the same network without using Batch Normalization and learning rate update on layer 4.
- Test accuracy decreased to 92% when we removed dropout from layer 4.
- Test accuracy decreased to 94.3% and 91.9%, when we changed dropout values to 0.5 and 0.8, respectively, in convolutional layers.

When we repeated the tests using the same architecture, different random parameter initialization resulted in slightly different results. For example, we obtained test accuracies like 96.3% and 95,6% on the same architecture and same number of training epochs.

To study the misclassified samples, we generated confusion matrices and saved all misclassified samples with an indication of original class and the assigned class, see Table 1. It was clear that most of the confusion comes from characters with similar morphology like “Daal” vs. “Raa” and “Zaay” ⟨ز⟩ vs. “Thaal” ⟨ذ⟩, or characters with diacritics (like dots) such as “Raa” ⟨ر⟩ vs. “Zaay” ⟨ز⟩ or “Jiim” ⟨ج⟩ vs. “Haa” ⟨ح⟩ and “Kha” ⟨خ⟩. This suggests the usefulness of another level of classifier system that is trained on classifying these shapes. Figure 8 illustrates some of the testing images that were misclassified due to the dot position (e.g. “Ghayn” ⟨غ⟩ to “Ayn” ⟨ع⟩), or to the number of dots (e.g “Qaaf” ⟨ق⟩ to “Faa” ⟨ف⟩), or to the curvature of the character (e.g. “Daal” ⟨د⟩ to “Raa” ⟨ر⟩).

4.2.2 AHCR Using AIA9K Dataset

We repeated the same set of experiments using our CNN architecture but on the AIA9K database. We divided the data into three parts; training, validation and testing, with ratios of 70%, 15% and 15% for each set, respectively. Then, we ran training for 17 epochs and obtained a classification accuracy of 93.4%. Changing dropout value from 0.5 to 0.75 caused test accuracy to reach 94.2%. Increasing the number of filters in a similar way to the approach described in sub-section 4.2.1 improved accuracy to 94.65% after 29 epochs. Finally, changing the fully connected layer dropout keep probability from 0.75 to 0.8 and training for 32 epochs improved test accuracy once more to reach 94.8%.

Table 1. Classification accuracy for each AHCD character followed by the count of each wrongly assigned character based on the confusion matrix.

Alif:714, (Acc=0.992), Miim:3, Kaaf:2, Daal:1
Baa:712, (Acc=0.989), Taa:3, Haa:2, Yaa:2, Kaaf:1
Taa:699, (Acc=0.971), Thaa:17, Nuun:3, Kaaf:1
Thaa:696, (Acc=0.967), Taa:13, Kaaf:6, Daad:2, Qaaf:2, Faa:1
Jiim:716, (Acc=0.994), Haa:4
Haa:707, (Acc=0.982), Kha:7, Jiim:6
Kha:708, (Acc=0.983), Haa:12
Daal:700, (Acc=0.972), Raa:10, Laam:3, Waaw:3, Thaal:2, Zaay:1, Kaaf:1
Thaal:688, (Acc=0.956), Daal:15, Zaay:7, Waaw:6, Thaa:1, Zhaa:1, Kaaf:1, Nuun:1
Raa:704, (Acc=0.978), Daal:7, Zaay:3, Waaw:3, Thaa:1, Kha:1, Kaaf:1
Zaay:661, (Acc=0.918), Thaal:31, Raa:24, Daal:2, Kha:1, Zhaa:1
Siin:717, (Acc=0.996), Saad:2, Daal:1
Shiin:716, (Acc=0.994), Qaaf:3, Siin:1
Saad:710, (Acc=0.986), Daad:5, Siin:2, Ha:2, Kaaf:1
Daad:693, (Acc=0.963), Saad:18, Shiin:4, Thaa:2, Faa:2, Kha:1
Taa:718, (Acc=0.997), Siin:1, Zhaa:1
Zhaa:690, (Acc=0.958), Taa:29, Ghayn:1
Ayn:691, (Acc=0.960), Ghayn:15, Miim:6, Haa:5, Kha:2, Taa:1
Ghayn:720, (Acc=1.000)
Faa:713, (Acc=0.990), Qaaf:4, Jiim:2, Kaaf:1
Qaaf:686, (Acc=0.953), Faa:25, Kaaf:4, Yaa:2, Taa:1, Jiim:1, Zaay:1
Kaaf:718, (Acc=0.997), Saad:1, Ha:1
Laam:715, (Acc=0.993), Zhaa:4, Daal:1
Miim:705, (Acc=0.979), Ha:9, Alif:5, Daad:1
Nuun:674, (Acc=0.936), Kaaf:18, Taa:15, Faa:7, Thaa:1, Raa:1, Daad:1, Zhaa:1, Qaaf:1, Yaa:1
Ha:706, (Acc=0.981), Waaw:13, Faa:1
Waaw:702, (Acc=0.975), Daal:7, Raa:6, Ha:5
Yaa:697, (Acc=0.968), Ayn:8, Haa:4, Faa:3, Jiim:2, Qaaf:2, Baa:1, Siin:1, Saad:1, Kaaf:1

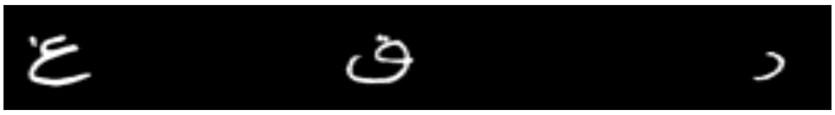


Figure 8. Misclassified test samples due to (i) dot position – "Ghayn" to "Ayn" (ii) Dot shape – "Qaaf" to "Faa" (iii) Character curvature – "Daal" to "Raa".

Similar to the tests performed in sub-section 4.2.1, adding an additional fully connected layer of the same size of the final fully connected layer decreased test accuracy to 94.3%. We tried different experiential adjustments to the network parameters to change the network characteristics, which didn't provide improvements.

Analysis of misclassified testing samples revealed similar observations to those in sub-section 4.2.1, but it also showed that some writers had actually written the "Haa" not in its isolated format ⟨ ه⟩ , but rather in its shape at the start of the word; initial form ⟨ هـ⟩ . In addition, some mislabeled characters were observed.

4.3 Comparison to Other Approaches

In this sub-section, the performance of our system is compared with those of other Arabic handwriting recognition systems that use AIA9K and AHCD datasets. Relevant results are presented in Table 2. The AIA9K and AHCD databases are available for the general public for research purposes and this makes system comparisons meaningful.

We concentrated our efforts on the largest databases; namely, AIA9K and AHCD. For AIA9K, our proposed system outperformed the system in [23] and was easier to implement and there was no need to evaluate many sets of engineered features and classifier combinations.

As for the AHCD database, our system was able to obtain a high percentage of accuracy of 97.6% as implemented using Keras/TensorFlow. The reported accuracy of the system in [19] is lower by 2.7%.

One of the features of working with Keras is that selection of hyper-parameters based on experts' opinions or thumb rules comes preprogrammed, so that programmers experiment with default parameters.

This demonstrates that our proposed algorithm is favorable to the state-of-the-art when we are dealing with the largest database available for AHCR. Our results are better than those of all of the experiments performed by Torki et al. [13]. Our design obtained better accuracy, since we started with a good architecture and added ritualization, tweaked the network parameters, increased the number of layers and adopted a data augmentation mechanism.

Table 2. Comparison between proposed approach and other approaches on the same datasets.

Authors	Database	Training Data & Testing Data	Classification Accuracy
Torki et al. [13]	AIA9k	8738 images 85% training 15% testing (by gender)	94.28%
El-Sawi et al. [19]	AHCD	16800 images 13440 Training images 3360 Testing images	94.9%
Proposed Approach	AIA9k	8738 images 85% training 15% testing	94.8%
Proposed Approach	AHCD	16800 images 13440 Training images 3360 Testing images	97.6%

We tried to build a famous network architecture called Residual Networks (ResNet) [31]-[32]. This network contains 18 layers and achieved an accuracy of 92% in the first few experiments. It is a promising architecture that we intend to study further.

4.4 Comparison to Latin Alphabet Classification

To study the misclassified characters, a comparison of the proposed network performance on Latin Alphabet (modern English characters) was deemed suitable to know if the network has the same failures as compared to the misclassification in Arabic characters. Does the error come from the proximity of the morphology of certain characters?

In this sub-section, we demonstrate the performance of our system on a database called EMNIST dataset, that is a set of handwritten character digits derived from the National Institute of Standards and Technology (NIST) Special Database 19 [40]. It has a dataset of Latin (English) alphabet that contains 145600 images of English alphabet (A-Z and a-z) sorted into 26 classes, where each class has both uppercase and lowercase characters converted into a 28x28 pixel image format. The dataset is divided into a training set of 124800 and a testing set of 20800 images.

We applied the same network architecture on that dataset and training was completed in 15 epochs with a test accuracy of 95%. This is far better than the best accuracy of 85% using 10,000 hidden layer neurons of the ELM network trained using the Online Pseudo-Inverse Update Method (OPIUM) reported in [40].

However, the goal of this experiment was merely to study whether the trained network will have problems with characters that have the same morphology. Table 3 lists the classification accuracy of each of the characters along with the number of times that each character was misclassified as another. It is obvious that the confusion is mostly seen between characters that have similar morphology, such as “l” and “I”, “G” and “Q” and “U’ and “V”. This confirmed our hypothesis that failures occurred in classifying Arabic characters like (“Alif” and “Miim”) or (“Baa”, “Taa” and “Thaa”) - See Table 1 - due to similar shape. Hence, data augmentation and increasing the network capacity were deemed a necessary to improve performance.

Table 3. Classification accuracy for each EMNIST character followed by the count of each wrongly assigned character based on the confusion matrix.

```
A:762, (Acc.=0.953), Q:6, D:4, F:4, H:4, N:4, C:3, O:3, U:3, Z:3, G:1, R:1, X:1, Y:1
B:787, (Acc.=0.984), E:2, H:2, Z:2, C:1, D:1, G:1, N:1, O:1, R:1, S:1
C:793, (Acc.=0.991), E:5, G:1, U:1
D:770, (Acc.=0.963), O:17, A:4, Q:3, B:1, C:1, J:1, N:1, P:1, T:1
E:785, (Acc.=0.981), C:9, L:2, A:1, I:1, P:1, W:1
F:784, (Acc.=0.980), T:8, E:2, G:2, P:2, I:1, R:1
G:664, (Acc.=0.830), Q:104, A:10, S:5, C:3, B:2, E:2, F:2, J:2, Y:2, D:1, N:1, O:1, P:1
H:765, (Acc.=0.956), N:17, L:5, B:4, K:3, X:2, R:1, T:1, U:1, W:1
I:558, (Acc.=0.698), L:223, J:10, C:2, E:2, R:2, B:1, O:1, V:1
J:756, (Acc.=0.945), I:22, D:5, F:3, S:3, T:3, X:2, Y:2, G:1, L:1, U:1, V:1
K:786, (Acc.=0.983), B:3, R:3, X:3, H:2, E:1, L:1, T:1
L:652, (Acc.=0.815), I:129, C:7, H:6, J:2, B:1, D:1, R:1, Y:1
M:794, (Acc.=0.993), N:4, K:1, W:1
N:780, (Acc.=0.975), M:5, R:5, H:3, W:2, X:2, I:1, U:1, V:1
O:784, (Acc.=0.980), D:10, A:2, U:2, C:1, Q:1
P:793, (Acc.=0.991), D:4, E:1, L:1, Q:1
Q:730, (Acc.=0.912), G:43, A:12, O:5, E:2, F:2, I:2, D:1, U:1, Y:1, Z:1
R:774, (Acc.=0.968), V:5, Y:5, K:4, T:3, X:3, C:1, E:1, I:1, N:1, O:1, Z:1
S:790, (Acc.=0.988), G:4, A:2, J:2, D:1, N:1
T:785, (Acc.=0.981), F:2, K:2, X:2, A:1, B:1, C:1, E:1, J:1, O:1, R:1, Y:1, Z:1
U:769, (Acc.=0.961), V:16, Y:3, C:2, N:2, W:2, A:1, H:1, J:1, L:1, O:1, S:1
V:736, (Acc.=0.920), U:36, Y:16, R:7, J:1, L:1, Q:1, X:1, Z:1
W:788, (Acc.=0.985), N:5, H:2, M:2, U:2, V:1
X:791, (Acc.=0.989), K:3, Y:2, H:1, N:1, P:1, V:1
Y:780, (Acc.=0.975), X:9, R:3, J:2, T:2, D:1, G:1, K:1, V:1
Z:795, (Acc.=0.994), C:1, F:1, I:1, J:1, L:1
```

4.5 Classification Performance on Form-based Arabic Characters

Since it was shown that most misclassification of the system was due to similarity in the shape of characters, it was deemed suitable to carry out some tests where we keep only one character of every group of letters that have the same form. For example, various consonants that have the same form (or major stroke), such as “Baa” 〈 ب〉 , “Taa” 〈 ت〉 and “Thaa” 〈 ث〉 and only distinguished by pointing diacritics (the number and location of dots). Other examples are "Jiim” 〈 ج〉 , “Haa” 〈 ح〉 and “Kha” 〈 خ〉 . In this sub-section, we keep only one sample of each of these groups of similar letters and study the classification accuracy. We kept only the letter “Baa” from the first group and the Letter “Haa” from the second group, …etc. The reduced alphabet of Arabic Letters by form is 16 characters listed in Table 4. The classification accuracy has increased to 98.22% as expected. Looking at Table 4 reveals that misclassification occurred again because writers tend to write “Daal” in a relaxed smooth curved shape that looks like “Raa”, but the opposite is not true. The other main source of misclassification is the loop in “Waa” that is misclassified as “Faa”. This suggests that a classifier system that classifies in two stages; the first classifies digits by form and the second tries to look at diacritics and other distinct features to carefully distinguish between characters of similar morphology, would prove useful.

5. CONCLUSIONS

Automated software systems for the recognition of Arabic characters could have huge applications in many industry and government sectors. In this paper, we presented a Deep Learning system based on convolutional neural network that is capable of classifying Arabic handwritten characters with a state-of-the-art classification accuracy of 94.8% and 97.6% on the AIA9k and AHDC datasets, respectively.

Table 4. Classification accuracy for each of the form-based Arabic character dataset followed by the count of each wrongly assigned character based on the confusion matrix.

Alif:719, (Acc.=0.999), Yaa:1
Baa:718, (Acc.=0.997), Ayn:1, Laam:1
Haa:704, (Acc.=0.978), Ayn:13, Alif:1, Baa:1, Siin:1
Daal:682, (Acc.=0.947), Raa:29, Waaw:6, Laam:3
Raa:715, (Acc.=0.993), Daal:3, Faa:1, Yaa:1
Siin:714, (Acc.=0.992), Baa:2, Saad:2, Ayn:1, Faa:1
Saad:712, (Acc.=0.989), Siin:8
Taa:719, (Acc.=0.999), Yaa:1
Ayn:713, (Acc.=0.990), Taa:2, Miim:2, Alif:1, Haa:1, Faa:1
Faa:717, (Acc.=0.996), Kaaf:2, Yaa:1
Kaaf:713, (Acc.=0.990), Yaa:3, Alif:2, Faa:1, Ha:1
Laam:715, (Acc.=0.993), Kaaf:4, Faa:1
Miim:695, (Acc.=0.965), Alif:12, Ha:7, Raa:2, Saad:2, Baa:1, Faa:1
Ha:692, (Acc.=0.961), Waaw:11, Faa:7, Taa:6, Saad:3, Siin:1
Waaw:668, (Acc.=0.928), Faa:27, Daal:9, Ha:9, Raa:4, Alif:1, Taa:1, Ayn:1
Yaa:719, (Acc.=0.999), Alif:1

The system employs some regularization techniques (Dropout and Batch Normalization), which provide performance and efficiency improvements. The system was implemented using TensorFlow and Keras framework.

As a future work, in addition to working more with ResNets, we plan to apply the algorithm on a larger and more diverse database. This could be done by merging more than one source of database. We plan to implement deeper networks with state-of-the-art techniques for regularization and optimization. In the next stages, we are planning to use segmentation-free techniques to be able to train our network with larger datasets that contain Arabic handwritten connected characters (words) to make use of the context using LSTM networks. Ultimately, this research will open the door to great opportunities expanding the applications of deep learning using these powerful libraries to problems of ancient Arabic handwritten character recognition.

ACKNOWLEDGEMENTS

This project is partially supported by grants of Hamdi Mango Center for Scientific Research (HMCSR) and the Deanship of Scientific Research at the University of Jordan, Amman, Jordan.

REFERENCES

[1] R. Plamondon and S. N. Srihari, "On-line and Off-line Handwriting Recognition: A Comprehensive Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 2, no. 1, pp. 63-84, 2000.

[2] A. Belaïd and N. Ouwayed, Segmentation of Ancient Arabic Documents, in "Guide to OCR for Arabic Scripts," Eds. Volker Märgner and Haikal El Abed, Springer-Verlag, London, pp. 103-122, 2011.

[3] G. Abandah, M. Khedher and K. Younis, "Handwritten Arabic Character Recognition Using Multiple Classifiers based on Letter Form," Proceedings of the 5th IASTED International Conference on Signal Processing, Pattern Recognition and Applications (SPPRA 2008), pp. 128-133, Innsbruck, Austria, 13-15 Feb. 2008.

[4] G. Abandah, M. Khedher and K. Younis, "Evaluating and Selecting Features for Recognizing Handwritten Arabic Characters," Tech. Report, Computer Eng. Dept., The Univ. of Jordan, Sep. 2007.

[5] G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Li and T. Hospedales, "When Face Recognition Meets with Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition," Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 142–150, Santiago, Chile, 13–16 December 2015.

[6] H. Cecotti, "Hierarchical K-nearest Neighbor with GPUs and a High-performance Cluster: Application to Handwritten Character Recognition," International Journal of Pattern Recognition and Artificial Intelligence, vol. 31, no. 2, pp. 1–24, 2017.

[7] M. Abadi et al. "Tensorflow: Large-scale Machine Learning on Heterogeneous Distributed Systems," arXiv preprint arXiv:1603.04467, 2016.

[8] F. Chollet and Keras, GitHub Repository, [Online], Available: https://github.com/fchollet/keras, GitHub, 2015.

[9] M. Elleuch, N. Tagougui and M. Kherallah, "Arabic Handwritten Characters Recognition Using Deep Belief Neural Networks," Proc. of the 12th International Multi-Conference on Systems, Signals & Devices (SSD15), pp. 1-5, Mahdia, Tunisia, 16-19 March 2015.

[10] H. Alwzwazy et al., "Handwritten Digit Recognition Using Convolutional Neural Networks," International Journal of Innovative Research in Computer and Communication, vol. 4, no. 2, 2016.

[11] A. Lawgali, "A Survey on Arabic Character Recognition, " International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 8, no. 2, pp. 401-426, 2015.

[12] J. Al Abodi and X. Li, "An Effective Approach to Offline Arabic Handwriting Recognition," Pattern Analysis and Applications, vol. 40, no. 6, pp. 1883-1901, 2014.

[13] M. Torki et al., "Window-based Descriptors for Arabic Handwritten Alphabet Recognition: A Comparative Study on a Novel Dataset," arXiv preprint arXiv:1411.3519, 2014.

[14] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 886-893, 2005.

[15] H. Bay, A. Ess, T. Tuytelaars and L. van Gool, "Surf: Speeded-up Robust Features," Journal of Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346-359, 2008.

[16] D. Lowe, "Distinctive Image Features F-ROM Scale-invariant Key Points," International Journal of Computer Vision, vol. 60, pp. 91–110, 2004.

[17] A. Lawgali et al., "HACDB: Handwritten Arabic Characters' Database for Automatic Character Recognition," European Workshop on Visual Information Processing (EUVIP), pp. 255-259, 2013.

[18] M. Elleuch et al., "Optimization of DBN using Regularization Methods Applied for Recognizing Arabic Handwritten Script," International Conference on Computational Science (ICCS 2017), vol. 108, pp. 2292-2297, Zurich, 12-14 June 2017.

[19] A. El-Sawy, M. Loey and H. El-Bakry, "Arabic Handwritten Characters Recognition Using Convolutional Neural Network," WSEAS Transactions on Computer Research, vol. 5, pp. 11-19, 2017.

[20] G. Fink, "1st International Workshop on Arabic Script Analysis and Recognition (ASAR 2017)," [Online], Available: http://asar.ieee.tn/speakers/, Nancy, France, April 3-5, 2017.

[21] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, pp. 335-339, [Online], Available: http://www.deeplearningbook.org.

[22] S. Loffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," [Online], Available: https://arxiv.org/abs/1502.03167v3, 2015.

[23] Dan C. Cireşan et al., "Handwritten Digit Recognition with a Committee of Deep Neural Nets on GPUs," arXiv preprint arXiv:1103.4487, 2011.

[24] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based Learning Applied to Document Recognition," Proceedings of IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] P. Golik, P. Doetsch and H. Ney, "Cross-entropy vs. Squared Error Training: A Theoretical and Experimental Comparison," Interspeech, vol. 13, 2013.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929-1958, 2014.

[27] D. P. Kingma and B. Adam, "A Method for Stochastic Optimization," [Online], Available: https://arxiv.org/abs/1412.6980.

[28] M. Pechwitz, S. S. Maddouri, V. Mrgner, N. Ellouze and H. Amiri, "Ifn/enit - database of Handwritten Arabic Words," Colloque Inter. Francophone sur lEcrit et le Document (CIFED), pp. 129–136, 2002.

[29] Y. Le Cun, K. Kavukvuoglu and C. Farabet, "Convolutional Networks and Applications in Vision," Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS'10), Paris, France, 2010.

[30] K. Younis and A. Alkhateeb, "A New Implementation of Deep Neural Networks for Optical Character Recognition and Face Recognition," Proc. of the New Trends in Information Technology (NTIT-2017), The University of Jordan, 25-27 April 2017.

[31] Ke Zhang, "Residual Networks of Residual Networks: Multilevel Residual Networks," IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Latex Class Files, vol. 14, no. 8, Aug. 2016.

[32] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, 2016.

[33] C. Boufenar, M. Batouche and M. Schoenauer, "An Artificial Immune System for Offline Isolated Handwritten Arabic Character Recognition. Evolving Systems," Springer-Verlag, pp.1-17, 2016.

[34] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807–814, Haifa, June 2010.

[35] J. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," arXiv preprint arXiv:1412.6806, 2014.

[36] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems (NIPS), pp. 1097-1105, 2012.

[37] I. Ahmad and G. Fink, "Class-based Contextual Modeling for Handwritten Arabic Text Recognition," International Conference on Frontiers in Handwritten Recognition (ICFHR), pp. 554-559, 2016.

[38] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," Journal of Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[39] B. Balci, D. Saadati and D. Shiferaw, "Handwritten Text Recognition Using Deep Learning," CS231n: Convolutional Neural Networks for Visual Recognition, Stanford Uni., Course Project Report, 2017.

[40] G. Cohen, S. Afshar, J. Tapson and A. Van Schaik, "EMNIST: An Extension of MNIST to Handwritten Letters," arXiv:1702.05373v2, 1 March 2017.

[41] A. Ray, S. Rajeswar and S. Chaudhury, "Text Recognition Using Deep BLSTM Networks," Proc. of the 8th International Conference on Advances in Pattern Recognition (ICAPR), pp. 1-6, Kolkata, India, 4-7 Jan. 2015.

ملخص البحث:

يمثل التحليل الآلي وتمييز الأحرف العربية المخطوطة باليد من الصّور مشكلة مهمة في كثير من التطبيقات. وعلى الرغم من التقدم الهائل في البحوث المرتبطة بتمييز الأحرف باستخدام الضوء في الآونة الأخيرة، فما زالت هناك بعض المشكلات التي تنتظر الحلّ، وبخاصة الأحرف العربية. حيث ان ظهور الشبكات العصبية العميقة يُعدّ أمراً واعداً جداً لإيجاد حلول لبعض تلك المشكلات. في هذه الورقة، نقدم شبكة عصبية عميقة لحلّ مشكلة تمييز الأحرف العربية المخطوطة باليد. تَستخدم الشبكة المقترحة نماذج التفافية من الشبكات العصبية مع متغيراتٍ خاصة بالتنظيم، مثل التسوية على دفعات، للحيلولة دون فَرْط الملاءَمة. وقد طبقنا الشبكة العصبية العميقة المقترحة على قواعد البيانات AIA9K وAHCD؛ إذْ كانت دقة التصنيف 94.8% و97.6% على الترتيب. وقد تمت دراسة أداء الشبكة على EMNIST ومجموعة البيانات AHCD المبنية على الشكل للمساعدة في التحليل.