IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 38, No. 1, February, 2024, pp. 1-9
Journal homepage: http://iieta.org/journals/ria

Data Augmentation for Offline Arabic Handwritten Text Recognition Using Moving Least Squares

Mohamed Amine Chadli[1], Rochdi Bachir Bouiadjra[1*], Abdelkader Fekir[1], Jesus Martínez-Gómez[2], José A. Gámez[2]

[1] Computer Science Department, University of Mustapha Stambouli, Mascara 29000, Algeria
[2] Computing Systems Department, Universidad de Castilla-La Mancha, Campus Universitario s/n, Albacete 02071, Spain

Corresponding Author Email: r.bachir-bouiadjra@univ-mascara.dz

https://doi.org/10.18280/ria.380101

Received: 19 July 2023
Revised: 3 November 2023
Accepted: 8 December 2023
Available online: 29 February 2024

Keywords:
off-line handwritten recognition, Arabic script, IFN/ENIT database, convolutional neural network, recurrent neural network, connectionist temporal classification, synthetic data, data augmentation

ABSTRACT

This paper addresses the research problem of Offline Arabic Handwriting Text Recognition (HTR). One of the most important approaches to HTR systems is deep learning. A large amount of annotated data is needed to train deep learning-based HTR systems. The Arabic language is spoken by hundreds of millions of people in North Africa and the Middle East. Writing styles and common words differ significantly between those regions. Due to the great diversity possible, designing a statistically represented and balanced database of Arabic handwritten texts by gathering and labeling the texts is an arduous task to achieve. One of the ways to enrich the training databases is by augmenting the existing data. We have developed a new data augmentation technique for Arabic handwritten texts using Moving Least Squares (MLS) to deform the images. This technique results in realistic images that look like manipulating real-world images, and the deformations are done using linear functions that produce deformations in real time. We aim to deform the training data images randomly in a way that the text present in the images is still recognizable by a human. This augmentation technique can be used directly on images to augment them unlike other techniques such as Generative Adversarial Networks (GAN) where they must be trained beforehand. At the same time, it produces new complex augmented images compared to simple traditional augmentation techniques such as rotations and translations. In addition to this augmentation technique, we used a deep learning system called Convolutional Recurrent Neural Networks (CRNN) to test the new technique, and we have experimented with a CRNN model that accepts small input-size images to boost the time needed for both training and image augmentations. All the experimentations are carried out on the Arabic IFN/ENIT database. The results show that the small input size CRNN model outperforms the large input size CRNN model by a big margin. The results also show that the integration of images augmented by the MLS technique can help the recognition system to generalize better on the test data, therefore, it can slightly improve the performance of the recognition system.

1. INTRODUCTION

The Arabic language is one of the most spoken languages in the world, and it is considered the official language of several countries. It is one of the six official languages of the United Nations (UN), and it is used in various official and unofficial documents by most Arabic speaking organizations. Organizations such as non-profits regularly conduct surveys that require filling out certain documents, to gather statistical information about a topic or region. Industries such as banking and healthcare routinely require their clients to write on documents. A common practice is to store documents in a digital format for fast, intelligent retrieval and automatic natural language processing operations, such as classification, clustering, and search engine ranking. Due to the big number of operations done daily, the process of reading these handwritten texts must be automated. For this reason, automatic Arabic handwriting text recognition systems are needed, and we aim to automatically recognize Arabic handwritten texts from photo images taken for Arabic texts. Thanks to this approach, we could eliminate the huge manpower that was usually needed to do this task manually and slowly, and speed up this task and make it more efficient.

Arabic HTR can be divided into two different types of recognition, online and offline text recognition. Recognizing the text after it was already written on some kind of medium (e.g., a sheet of paper, tablet, official form), with the condition that the writing was already complete, is known by the name of offline text recognition. On the other hand, recognizing the text in real-time during writing on some kind of medium like a tablet using a digital pen is known as online text recognition. Historically speaking, early attempts to create optical character recognition (OCR) systems focused solely on recognizing predefined shapes of individual characters,

meaning that OCR systems were only capable of recognizing one type of font up to a maximum of 10 font types depending on the methods applied. Which led to the first generation of commercially available OCR systems for Latin scripts between 1960 and 1965. Yet, it took 30 years for the Arabic OCR systems to be available at the market in the 1990's, due to the connectedness nature of Arabic words. A lot of details are presented by Iman Yousif et al. in their review [1] on Arabic HTR. Additionally, the systems that can be used to recognize Arabic HTR can be divided into two forms of systems, segmentation-based and segmentation-free systems. Segmentation-free approaches are also known as the holistic approaches, where each word or text is considered a class, and the system classifies each image into a class, in this case, a word or a text. Thus, the task of recognizing Arabic HTR is transformed into a classification problem. Consequently, this approach will inherit all the advantages and disadvantages of traditional classification systems. The main advantage is that the implementation of this type of system is straightforward compared to segmentation-based approaches, and one of the disadvantages is that the number of classes in these Arabic HTR systems will heavily depend on the number of words or texts present in the specific domain of recognition. We must also take into consideration that the words to be recognized will be very limited to the specific domain, and these systems will not be able to recognize any word that is not present in the specified domain. For this reason, these types of systems will perform better on constrained domains. The segmentation-based approaches [2-8] rely on recognizing the characters present in the image one by one, instead of recognizing the text or the word as a whole. As we are trying to recognize the characters, this kind of system's class count is determined by the total amount of characters of the language in which the texts are written. Moreover, segmentation-based approaches are further subdivided into explicit and implicit segmentation methods. Images in the training dataset must be annotated at the character level in order to use explicit segmentation techniques. Each training image in the dataset should be segmented into sub-images, where each sub-image contains a character. While implicit segmentation approaches only require word-level annotations, (there is no need to segment the image into character sub-images during training) the system will try to recognize the characters implicitly. One notable advantage inherent in segmentation-based approaches lies in their adaptability across various domains. These systems possess the theoretical capacity to recognize any given word or line of text. Notably, implicit segmentation approaches offer an additional advantage over explicit approaches, necessitating substantially fewer labeled datasets for training compared to their explicit counterparts, rendering explicit segmentation-based approaches more challenging to train, as previously elucidated. Another thing to consider is that implementing segmentation-based systems is relatively difficult. Al Abodi et al. [6], Parvez and Mahmoud [7] and Elzobi et al. [8] worked with explicit segmentation strategies. While Jayech et al. [2], Amrouch et al. [3], El-Hajj et al. [4], and El Moubtahij et al. [5] used implicit segmentation strategies in their recognition systems. Segmentation-based approaches are also known as the analytic approach, and they tend to perform better than the segmentation-free approaches when we have a large vocabulary of words to be recognized. Other researchers have taken a different route, enhancing the recognition systems performance only by incorporating newly synthesized or augmented images to the training datasets. Recent research has focused on generating new synthesized augmented images using generative adversarial networks (GAN), which use two competing neural networks, one of the networks having the role of image generator and the other, the role of discriminator. The generator network tries to generate images as authentic as possible that look exactly like real images, and the discriminator network tries to determine whether the generated image is authentic or not. GANs have been proven effective in several areas of computer vision for generating fake images that appear authentic. For instance, Saabni et al. [9] developed a simple system that generates a synthetic dataset of Arabic words in a specific lexicon. They generate the word parts using a predefined large set of shapes of each Arabic character in each position. Graves [10] has developed an approach to generate new synthetic datasets for handwriting texts using Recurrent Neural Networks. This system generates highly realistic cursive handwriting texts in a wide variety of styles. Elarian et al. [11] adopted two concatenation models (Extended-Glyphs connection and Synthetic-Extensions connection) to synthesize Arabic words from segmented characters. Wigington et al. [12] have introduced two data augmentation and normalization techniques, which are used with a CNN-LSTM. Alonso et al. [13] proposed a system based on Generative Adversarial Networks (GAN) to generate new synthesized images for French and Arabic datasets. In their implementation of the GAN, they add an auxiliary network for text recognition, where they train the system with a balanced combination of an adversarial loss and a Connectionist Temporal Classification (CTC) loss. Jha et al. [14] used a Generative Adversarial Network (GAN) to augment data text images and experimented on several handwritten digits (Latin, Bangla, Devanagri, and Oriya). They also experimented with adding too many artificial augmented samples. Fogel et al. [15] followed a semi-supervised approach to generate images of handwritten words with arbitrary lengths using Generative Adversarial Networks (GAN).

In this paper, we have developed a variation of Convolutional Recurrent Neural Networks (CRNN) [16], which is an implicit segmentation-based approach. An overview of the CRNN system is presented in Figure 1. Our modification proposal allows to use small images as input. These CRNN systems consist of three main blocks: a convolutional block, a recurrent block, and a transcription block. The transcription block is a Connectionist Temporal Classification (CTC) layer [17], that deals with unsegmented sequence inputs. We have also applied some traditional augmentation techniques to the Arabic IFN/ENIT database [18]. We finally concluded our research by developing a new text augmentation technique that is best suited to handwritten Arabic texts. This technique is based on the Moving Least Squares (MLS) [19] by Schaefer et al. to generate new images for training.

2. METHODOLOGY

Many different methods have been used in the research area of Arabic HTR. Some of these popular methods is deep learning-based methods. In this work, we have initially used a model called CRNN, where we employed the outcomes derived from the adaptation of the CRNN model, which was initially introduced in the study [20]. We used this model as a baseline. This model possesses the capability to accommodate

sizable images as input, thereby enabling us to investigate the hypothesis asserting that superior performance can be achieved by utilizing larger input-size images. Second, we have designed another variation of the CRNN model that can accept smaller input-size images, aiming to compare its performance with the former model. Third, we have supplemented the second CRNN model (the one that accepts small input-size images) with new images augmented from the chosen dataset by traditional augmentation techniques. This was done to check the effects of employing these new augmented images on our Arabic HTR model. Lastly, we have developed a new augmentation technique based on the MLS technique that better suits our case of study to recognize Arabic handwriting texts. All the details of the previously mentioned methods are discussed in the following subsections.

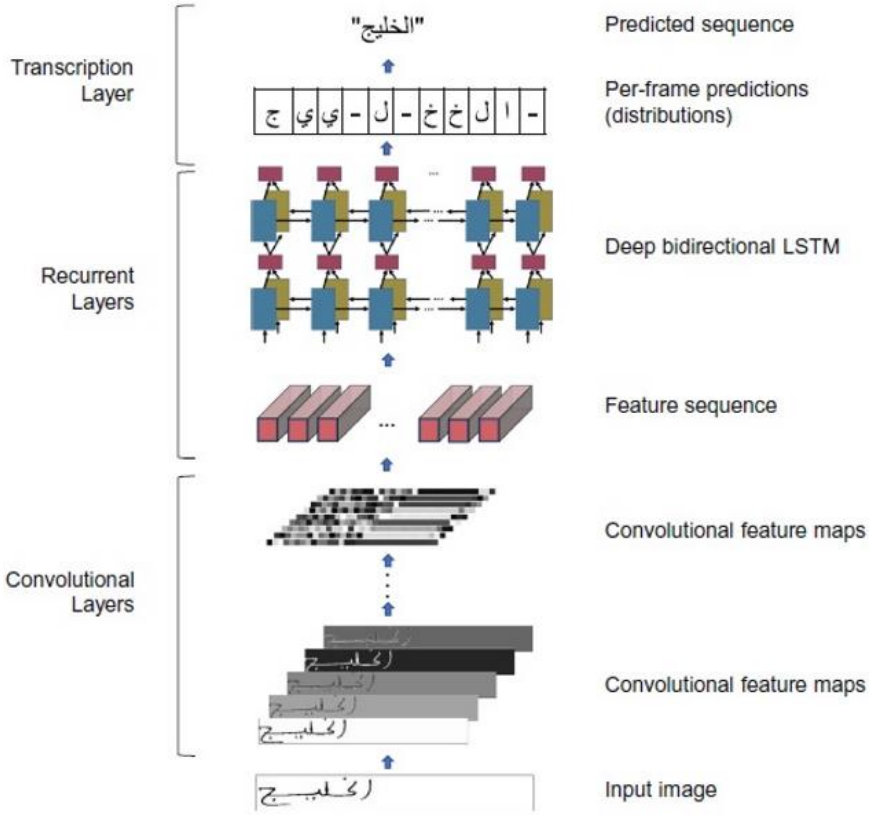

Figure 1. Network configuration. The network has three components: 1) convolutional layers, which use the image as input to create sequences of features; 2) recurrent layers, which generate feature maps with context; 3) transcription layer, which converts a final label sequence from the character sequence

Notes: Reprinted from the study [20]. Reprinted with permission.

2.1 Large input-size and small input-size models

The network architecture of the two models follows the same CRNN flow, they just differ in the size of the input images they accept, the number of layers, and some kernel and stride sizes for some layers. The large input size model and the small input size model architectures are described in detail respectively in Tables 1 and 2. The networks are defined in the tables starting at the lowest point and moving upward. The first step involves normalizing the input images after scaling them to (64, 512), (32, 128), respectively. We used convolutional layers for the first two layers, and then the Rectified Linear Unit (ReLU) activation function. To expedite training, the BatchNormalization layer is applied after every convolutional layer save for the initial layers. Our models were designed using a VGG-like approach, with 3 × 3 convolution kernels and a gradual increase in feature maps from 64 to 512. The last convolutional layer in both models use 2 × 2 convolution kernels. To minimize the size of the feature maps, we additionally applied 2 × 2 MaxPooling layers following the convolutional layers, with the exception of the last MaxPooling layer in the large input size model, and the last two MaxPooling layers in the small input size model, where we used a 2 × 1 MaxPooling window and a 2 × 1 stride. The prescribed adjustments were implemented to produce wide feature maps, allowing us to predict every Arabic word. Consequently, the resultant output of the CNN block is a sequence of 31 vectors of feature maps. These 31 vectors adequately enable the prediction of all Arabic words present in the language. The choice of number of layers, their order, kernel sizes, strides and padding sizes, was specifically made to design an architecture that produces a sequence of 31 feature vectors to be used later to predict the sequence of characters present in the image. If we randomly choose the previous configuration, we will get an architecture that produces a different number of sequence vectors or no sequence at all, which does not conform to our specified requirements. To adapt the CNN block's output to the proper shape of the recurrent layers input, the network features a unique layer called "MapToSequence." It will then be fed to the recurrent layers as a sequence of feature vectors, $X = [x_1, x_2, \cdots, x_T]$, where the recurrent layers use each vector x_t from the sequence vectors to create a new contextual feature vector, y_t. $Y = [y_1, y_2, \cdots, y_T]$ is the resulting sequence of contextual feature vectors. Fully connected layers are positioned after the recurrent layers in order to predict the label

distribution. The CTC part of the model requires these fully connected layers. In our case, the Arabic characters represent the labels that we are attempting to recognize. One character prediction is made for each contextual feature vector. The contextual feature vectors were extracted through the use of deep Bidirectional Long Short-Term Memory (Bi-LSTM) recurrent layers. In the original image, as seen in Figure 2, certain characters cover multiple neighboring regions.

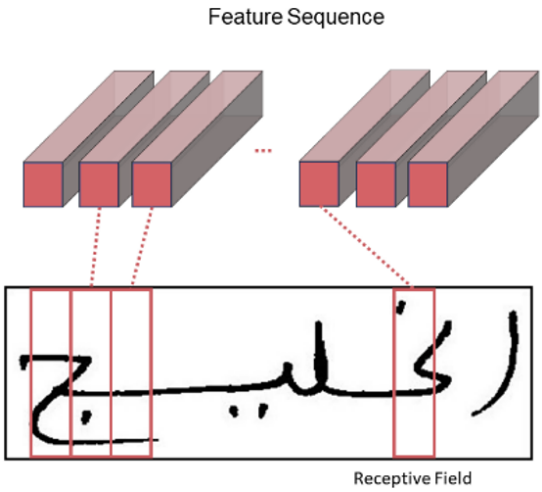

Figure 2. The area of receptivity. The original input image's regions are represented by each receptive field
Notes: Reprinted from [20]. Reprinted with permission.

Because of this, the non-contextual sequential features generated by the CNN layers are not the ideal choice when attempting to recognize characters that are stretched across multiple receptive fields. To extract features with context, we used RNNs; however, since vanilla RNNs are known to be inefficient at capturing long-term contextual information, we specifically used LSTM layers for this purpose. Utilizing Bi-LSTM, we were also able to obtain contextual information from the past as well as the future, which is important for text recognition because of its nature. Additionally, we achieved a deep extraction of contextual information by stacking two Bi-LSTM layers on top of each other. Figure 3 is a representation of a deep Bi-LSTM network.

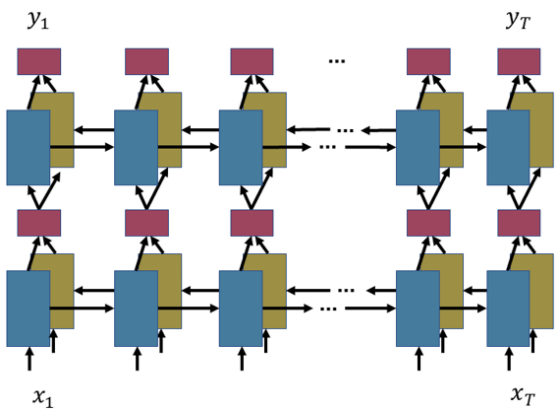

Figure 3. Our architecture employs a deep bidirectional LSTM, which combines forward (from left to right) and backward (from right to left) LSTMs to produce a bidirectional structure. Our deep bidirectional LSTM is constructed by vertically stacking multiple bidirectional LSTMs
Notes: Reprinted from the study [20]. Reprinted with permission.

Table 1. Network configuration summary of Large Input-Size Model. The uppermost row denotes the top layer. Within this context, ‘s’, ‘k’, ‘d’, and ‘p’ correspond to strides, Kernel size, dropout, and padding size, in that order. The variable "chars" represents the count of characters within the language constituting the dataset

Type	Configurations
Transcription	-
Dense	#units: chars + 1, activation: softmax, regularization: L2
Bi-LSTM	#hidden units: 128, d: 0.2
Bi-LSTM	#hidden units: 128, d: 0.2
Map-To-Sequence	-
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 512, K: 2 × 2, S: 1, P: Valid
MaxPooling	Window: 2 × 1, S: 2 × 1
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 128, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 128, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
Convolution	#maps: 64, K: 3 × 3, S: 1, P: Same
Activation	ReLU
Convolution	#maps: 64, K: 3 × 3, S: 1, P: Same
Input	64 × 512 black and white image-

Notes: Reprinted from the study [20]. Reprinted with permission.

A dropout of 0.2 was set on the recurrent layers and the weights of these layers were optimized with backpropagation through time algorithm. Finally, the output of the recurrent layers is fed to the transcription layer. The transcription layer functions akin to a translator, using the CTC technique [17] for training and inference on unsegmented data utilising recurrent layers. The CTC technique converts the sequence of letters predicted by the recurrent layers into a true sequence of labels. For the CTC method to work properly, the output of the Recurrent layers must be softmax with one more unit than there are labels (Characters) in the targeted dataset. The activation of the first units represents the probability of predicting that particular label (Character) at that specific timestep. The activation of the newly added unit means a ‘blank’, or no label. Jointly, these softmaxed outputs represent the probabilities of all possible ways of aligning all the

possible label sequences given an image input to the network. The network will be trained with a CTC loss function as described in the original work.

Table 2. Network configuration summary of Small Input-Size Model. The uppermost row denotes the top layer. Within this context, 's', 'k', 'd', and 'p' correspond to strides, Kernel size, dropout, and padding size, in that order. The variable "chars" represents the count of characters within the language constituting the dataset

Type	Configurations
Transcription	-
Dense	#units: chars + 1, activation: softmax, regularization: L2
Bi-LSTM	#hidden units: 128, d: 0.2
Bi-LSTM	#hidden units: 128, d: 0.2
Map-To-Sequence	-
Convolution	#maps: 512, K: 2 × 2, S: 1, P: Valid
MaxPooling	Window: 2 × 1, S: 2 × 1
BatchNormalization	-
Activation	ReLU
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
BatchNormalization	-
Activation	ReLU
Convolution	#maps: 512, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 1, S: 2 × 1
Activation	ReLU
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 256, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 128, K: 3 × 3, S: 1, P: Same
MaxPooling	Window: 2 × 2, S: 2
Activation	ReLU
BatchNormalization	-
Convolution	#maps: 64, K: 3 × 3, S: 1, P: Same
Activation	ReLU
Input	64 × 512 black and white image-

2.2 Traditional augmentation techniques

We have followed a traditional data augmentation technique to deal with an unbalanced dataset, as some words are really under-represented. We have augmented the dataset and generated new images by applying four main augmentation strategies that can suit the sequential nature of Arabic texts. First, we rotated the images by 4 degrees counter-clockwise direction around its center. Second, we have rotated the images by another 4 degrees but this time in a clockwise direction around its center. Third, we have dilated the text in the images by reinforcing the thickness of the text that appears in them. Fourth and last, we have translated the texts in images to the left by adding white pixels to the left of the images. An example of these augmentations is shown in Figure 4.

2.3 The proposed data augmentation method

To generate new augmented images for our training, we have used a technique of image deformation based on Moving Least Squares (MLS). This technique was first described in the study [19], and we specifically used the similarity deformation transformation because it best suits the case of Arabic handwritten texts. The newly generated deformed images are either slightly different from the original images or very different from them, depending on the applied deformation parameters to obtain these new deformed images. For these deformations to take place, the user must first select some parameters, the parameters are a set of fiducial points that will be moved to new positions.

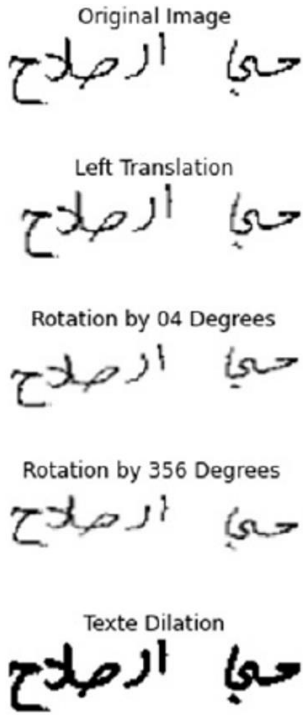

Figure 4. An example of the four traditional augmentation techniques: Original image, left translation, rotation by 04 degrees, rotation by 356 degrees, and text dilation

We randomly select these fiducial points p by sampling them from a discrete uniform distribution of the size of the images. The fiducial points to be sampled are equal to $4L$, where L is the length of the sequence of characters present in the current image that we want to augment from the training set. After that, we augment the images by randomly moving the fiducial points p to the new coordinates q within a random radius of R. To get the new augmented image, we apply the similarity deformation based on Moving Least Squares (MLS) on an input image. Given a point v from the input image, the transformation for this v based on the fiducial points p when moved to the new positions q is

$$T_v(q) = (v - p_*)M + q_* \quad (1)$$

where, M is a linear transformation matrix of a constant size of 2 × 2, and it is constrained to have the property of $M^T M = \lambda^2 I$ for some scalar λ. The p_* and $q_*$are the weighted centroids of the initialized fiducial points p and q respectively

$$p_* = \frac{\sum_{i=1}^{4L} w_i p_i}{\sum_{i=1}^{4L} w_i}, q_* = \frac{\sum_{i=1}^{4L} w_i q_i}{\sum_{i=1}^{4L} w_i}, \quad (2)$$

In the above equations, p_i and $q_i$are row vectors and the weights w_i for the point v have the form

$$w_i = \frac{1}{|p_i - v|^{2a}}, p_i \neq v \quad (3)$$

We can observe that when v approaches p_i, w_i increases significantly, which means that $T_v(p_i) = q_i$. And it also means that the points close to p_i will be also close to the new point q_i in the deformed image. Note that, if $p_i = q_i$, then each $T_v(v) = v$ for all v and, thus, T is the identity transformation function. We obtain the best transformation $T_v(v)$ by minimizing

$$\sum_{i=1}^{4L} w_i |T_v(p_i) - q_i|^2 \quad (4)$$

3. EXPERIMENTS

All experimentation details about the dataset, the role of image sizes, the role of traditional data augmentation techniques, and the role of the newly proposed data augmentation technique to improve recognition performance are included in the following subsections.

3.1 Dataset details

All experiments were carried out on the IFN/ENIT database, developed by Pechwitz et al. [18]. The database contains 26,400 images of handwritten Tunisian Arabic town/village names. These represent more than 210,000 characters. It was created by 411 distinct writers and divided into seven sets (a, b, c, d, e, f, and s), all of which were written by Tunisians with the exception of set s, which was written by UAE-based writers.

3.2 Training details of small input-size and large input-size models

The images contained in the Tunisian dataset are of significant dimensions, typically exceeding 1000 pixels in width and surpassing 200 pixels in height. Consequently, we postulate that diminishing the size of these images through resizing to smaller dimensions would result in a decline in the model's performance. Consequently, we obtained the outcomes using the CRNN model outlined in the study [6], which preserved the original large image sizes. Specifically, all images were resized to dimensions of (64, 512), representing a substantial input size.

We have also tested the opposite hypothesis that presumes that reducing the size of the images will not affect the performance of the model. For this, we have resized all the images to the size (32, 128). We have tested this hypothesis by designing another variation of the CRNN model that can accept images of smaller sizes.

We developed our networks in TensorFlow v2, using customised implementations of the MapToSequence layer, the transcription layer, and the BK Tree data structure (written in Python 3). The large input size model has 9,186,105 parameters, whereas the small input size model has 6,634,617 parameters. Because of the large input size of the first model, we need eight (08) times more memory space for both the ram and the GPU ram compared to the second small input size model to be able to train the first large input size model. We trained our models using pairs of training images and the corresponding ground-truth label sequences, with a batch size of 256 and two types of GPU: Tesla P100 with 16GB of RAM and Tesla T4 with 16GB of RAM. We stopped training when reaching convergence at epoch number 53 for the large input size model, while the small input size model reached convergence at epoch number 19. For the model with a large input size, we optimised using the Stochastic Gradient Descent (SGD) algorithm with a momentum of 0.9 and a learning rate of 0.01. For the smaller input size model, we utilized the Adam optimizer with a learning rate of 0.001. The training for the larger input size model lasted two hours, with an average of 126 seconds for every epoch. It took around 36 minutes, with 114 seconds for each epoch for the small input size model.

3.3 Training details of the CRNN model with traditional augmented images

All four augmentations are applied to the dataset images one after another, which means that the size of the dataset has increased five (5) times, and the model will receive during training five different versions of the same image consecutively. The first image without augmentation, the second one with a rotation of 4 degrees counter-clockwise direction, the third one with a rotation of 4 degrees clockwise direction, the fourth one with dilation of the text, and lastly the fifth one with a left translation of the text. The experimentations of these augmentation strategies are carried out on the smaller input-size CRNN model.

3.4 Training details of the CRNN model with MLS augmented images

We have applied the MLS technique to generate three new augmented images for each image in the dataset. Firstly, we randomly select the control points p for each image that we want to augment. Then, we randomly chose the new positions q for the augmentation to take place. Each point in the control points p will be randomly selected from a discrete uniform distribution of the values in the half-open interval [0, Image Width), and [0, Image Height) to obtain the width and height coordinates of the point. These new positions q are within a radius R from the original control points p. The distance between the control points coordinates of p and the corresponding destination points coordinates of q will determine how much deformation will be applied to the image of interest. As long as the distance between the p and the corresponding q is close, the newly generated image will keep a lot of its original meaning and context. The inverse holds true, provided that there is a signicant distance between them, the newly generated image will lose a lot of its original meaning and context. We manually tested several values of the radius R and visually observed which values would be effective for our case study. When we tested with large values, the deformation result was not good and the text present in the image was completely unreadable to the human eye, that's why we decided to choose the values that do not result in a total loss of readability of the text. We deform each image of interest three times, one with a small value of the radius R, another one with a medium value of the radius, and the last one with a relatively big value of it. We named them based on their deformation effect on the image of interest respectively, soft deformation, medium deformation, and hard deformation. Some examples of images augmented with these deformations are shown in Figure 5.

We kept in consideration that all the previous deformations will not affect the overall meaning and context of each original image, for which a human can still be able to read the content of each image correctly. After generating the three new images based on the three deformation variations, the soft, the medium,

and the hard one, we feed them along the original undeformed image to the small input size version of the CRNN model and train with them until convergence.

Figure 5. Three examples of augmented images using the MLS augmentation technique with: Soft deformation, medium deformation, and hard deformation

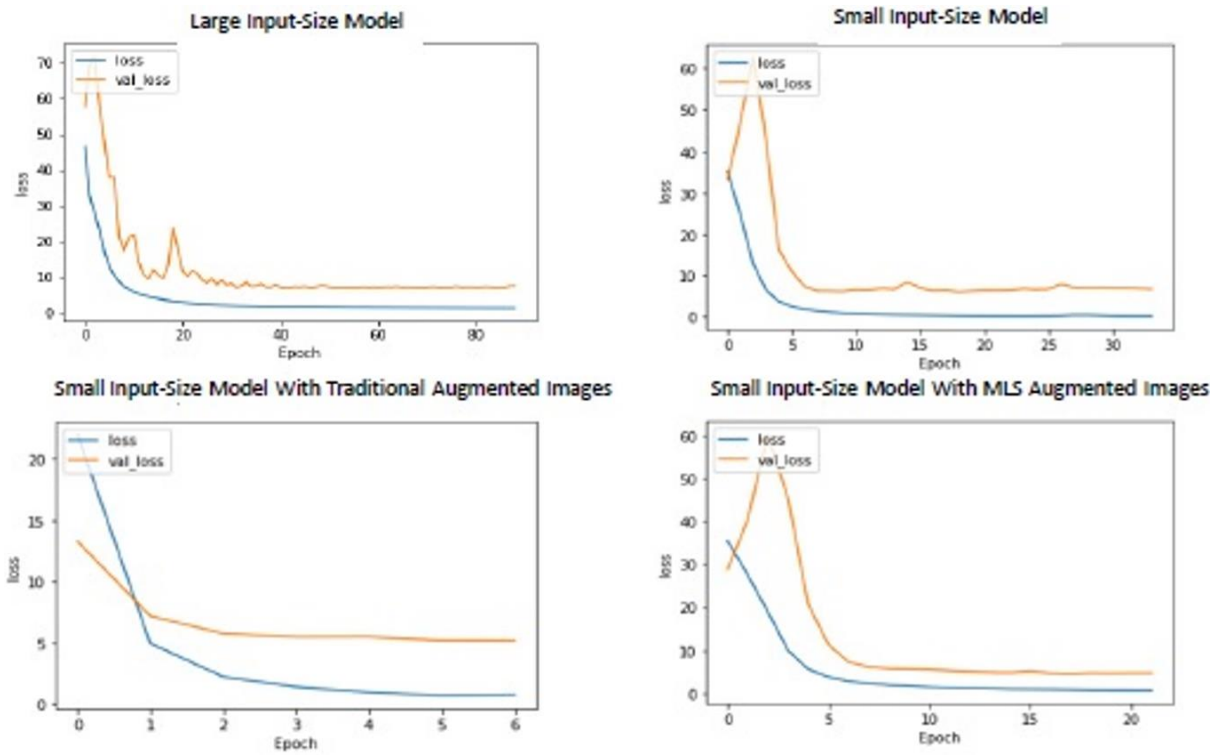

Figure 6. Losses during training and validation as a function of epoch count

In Figure 6 we respectively plot the training loss, and the validation loss of our models (Large Input Size Model, Small Input Size Model, Small Input Size Model with Traditional Augmented Images, and Small Input Size Model with MLS Augmented Images) as a function of the number of epochs.

3.5 Results and discussion

Numerous research teams have carried out experiments on the IFN/ENIT database using various training scenarios. Each team implemented its unique training strategy. In our case, we have opted to train the model on sets a, b, c, and d, while conducting validation on set e and performing testing on set f. This selection is motivated by the scenario's popularity and its alignment with our specic requirements, making it the most suitable and convenient option. The results of all experiments are shown in Table 3. The calculation of the accuracy is not straightforward, because of the nature of the predictions made by our models, which predict a sequence of characters instead of just one class. For this reason, we come up with this way of calculating the accuracy: if the model makes any mistake in any of the predicted characters of the predicted text, we simply consider the predicted text to be wrong. Otherwise, the predicted text is considered a true label prediction. We further reinforce the performance of our models by comparing the predicted texts with a dictionary of accepted predictions in a specific domain (in this case, the Tunisian city names) with the BK-Tree data structures.

Table 3. Accuracy of word recognition in training, validation, and testing conducted on the IFN/ENIT database for all experiments

IFN/ENIT Dataset Version: v2.0p1e					
				Accuracy %	
	Sets	Large Input-Size Model	Small Input-Size Model	Small Input-Size Model with Traditional Augmented Images	Small Input-Size Model with MLS Augmented Images
Training	a,b,c, d	99.9	99.9	99.9	99.9
Validation	E	80.97	81.02	**81.17**	80.82
Testing	F	79.95	81.54	80.59	**81.81**

We can see that the large input size model has the worst performance on the test set f, whereas the small input size model has outperformed it by a big margin of 1.59%. As a result, we can see that the assumption that the large input size model would outperform the small input size model was wrong. The opposite was correct. We can also see that the model that employs the MLS technique to augment the images has performed the best on the test set f, compared to other experiments, where it outperformed the small input size model with a small margin of 0.27%. Table 4 displays the comparative outcomes of prior systems.

Table 4. Accuracy of word recognition across many systems that were evaluated on set e and trained on sets a, b, c, and d

System	Accuracy %
Parvez and Mahmoud (2013) [7]	79.58
CNN-HMM (Amrouch et al. 2018 [3])	89.23
Alkhateeb et al. (2011) [21]	DBN (66.65) HMM (82.3)
El Moubtahij et al. (2017) [5]	78.95
Hamdani et al. (2009) [22]	81.93
Kessentini et al. (2012) [23]	79.6
DBN (Jayech et al. 2016) [24]	78.5
Our model with MLS Augmented Images	**81.17**

4. CONCLUSION

In this study, we developed a CRNN variant that accepts small input images to recognize offline Arabic handwritten text. We compared its performance with a large input size CRNN model and observed that the small input size model performed better. Therefore, our conclusion suggests that the adoption of large input-size models may not be imperative. Instead, employing a small input size model offers reduced memory space by a factor of 8, nearly quadruple faster convergence during training, and a 27% reduction in number of parameters, while in the meantime benefiting from better performance.

Moreover, we explored traditional augmentation techniques and introduced a novel MLS-based augmentation technique for experimentation. Our proposed augmentation technique exhibited promising results on the test set in comparison to other experiments. Further research is recommended to investigate the possible limitations of this novel augmentation technique and to leverage its capabilities in the development of new data augmentation strategies tailored specifically for offline Arabic handwritten texts.

REFERENCES

[1] Yousif, I., Shaout, A. (2014). Off-Line handwriting Arabic text recognition: A survey. International Journal of Advanced Research in Computer Science and Software Engineering, 4(9): 68-82.

[2] Jayech, K., Mahjoub, M.A., Amara, N.E.B. (2016). Synchronous multi-stream hidden markov model for offline Arabic handwriting recognition without explicit segmentation. Neurocomputing, 214: 958-971. https://doi.org/10.1016/j.neucom.2016.07.020

[3] Amrouch, M., Rabi, M., Es-Saady, Y. (2018). Convolutional feature learning and CNN based HMM for arabic handwriting recognition. In: Mansouri, A., El Moataz, A., Nouboud, F., Mammass, D. (eds) Image and Signal Processing. ICISP 2018. Lecture Notes in Computer Science, vol 10884. Springer, Cham. https://doi.org/10.1007/978-3-319-94211-7_29

[4] El-Hajj, R., Likforman-Sulem, L., Mokbel, C. (2005). Arabic handwriting recognition using baseline dependant features and hidden Markov modeling. In Eighth International Conference on Document Analysis and Recognition (ICDAR'05), Seoul, Korea (South), pp. 893-897. https://doi.org/10.1109/ICDAR.2005.53

[5] El Moubtahij, H., Halli, A., Satori, K. (2017). Using features of local densities, statistics and HMM toolkit (HTK) for offline Arabic handwriting text recognition. Journal of Electrical Systems and Information Technology, 4(3): 387-396. https://doi.org/10.1016/j.jesit.2016.07.005

[6] Al Abodi, J., Li, X. (2014). An effective approach to offline Arabic handwriting recognition. Computers & Electrical Engineering, 40(6): 1883-1901. https://doi.org/10.1016/j.compeleceng.2014.04.014

[7] Parvez, M.T., Mahmoud, S.A. (2013). Arabic handwriting recognition using structural and syntactic pattern attributes. Pattern Recognition, 46(1): 141-154. https://doi.org/10.1016/j.patcog.2012.07.012

[8] Elzobi, M., Al-Hamadi, A., Al Aghbari, Z., Dings, L., Saeed, A. (2014). Gabor wavelet recognition approach for off-line handwritten Arabic using explicit segmentation. In: S. Choras, R. (eds) Image Processing and Communications Challenges 5. Advances in Intelligent Systems and Computing, vol 233. Springer, Heidelberg. https://doi.org/10.1007/978-3-319-01622-1_29

[9] Saabni, R.M., El-Sana, J.A. (2013). Comprehensive synthetic Arabic database for on/off-line script recognition research. International Journal on Document Analysis and Recognition (IJDAR), 16(3): 285-294. https://doi.org/10.1007/s10032-012-0189-5

[10] Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850. https://doi.org/10.48550/arXiv.1308.0850

[11] Elarian, Y., Ahmad, I., Awaida, S., Al-Khatib, W.G., Zidouri, A. (2015). An Arabic handwriting synthesis system. Pattern Recognition, 48(3): 849-861. http://dx.doi.org/10.1016/j.patcog.2014.09.013

[12] Wigington, C., Stewart, S., Davis, B., Barrett, B., Price, B., Cohen, S. (2017). Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network. 2017 14th IAPR International Conference on Document analysis and Recognition (ICDAR), Kyoto, Japan, pp. 639-645. https://doi.org/10.1109/ICDAR.2017.110

[13] Alonso, E., Moysset, B., Messina, R. (2019). Adversarial generation of handwritten text images conditioned on sequences. 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, NSW, Australia, pp. 481-486. https://doi.org/10.1109/ICDAR.2019.00083

[14] Jha, G., Cecotti, H. (2020). Data augmentation for handwritten digit recognition using generative adversarial networks. Multimedia Tools and Applications, 79: 35055-35068. https://doi.org/10.1007/s11042-020-08883-w

[15] Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R. (2020). Scrabblegan: Semi-supervised varying length handwritten text generation. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 4323-4332. https://doi.org/10.1109/CVPR42600.2020.00438

[16] Shi, B., Bai, X., Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(11): 2298-2304. https://doi.org/10.1109/TPAMI.2016.2646371

[17] Graves, A., Fernández, S., Gomez, F., Schmidhuber, J. (2006). Connectionist temporal classication: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning, pp. 369-376. https://doi.org/10.1145/1143844.1143891

[18] Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H. (2002). IFN/ENIT-database of handwritten Arabic words. In Proceeding of CIFED, pp. 127-136.

[19] Schaefer, S., McPhail, T., Warren, J. (2006). Image deformation using moving least squares. In ACM SIGGRAPH 2006 Papers, pp. 533-540. https://doi.org/10.1145/1179352.1141920

[20] Chadli, M.A., Bachir Bouiadjra, R., Fekir, A. (2023). Offline arabic handwritten text recognition for unsegmented words using convolutional recurrent neural network. In: Salem, M., Merelo, J.J., Siarry, P., Bachir Bouiadjra, R., Debakla, M., Debbat, F. (eds) Artificial Intelligence: Theories and Applications. ICAITA 2022. Communications in Computer and Information Science, vol 1769. Springer, Cham. https://doi.org/10.1007/978-3-031-28540-0_22

[21] AlKhateeb, J.H., Pauplin, O., Ren, J., Jiang, J. (2011). Performance of hidden Markov model and dynamic Bayesian network classifiers on handwritten Arabic word recognition. Knowledge-Based Systems, 24(5): 680-688. https://doi.org/10.1016/j.knosys.2011.02.008

[22] Hamdani, M., El Abed, H., Kherallah, M., Alimi, A.M. (2009). Combining multiple hmms using online and o-line features for off-line Arabic handwriting recognition. In 2009 10th International Conference on Document Analysis and Recognition, Barcelona, Spain, pp. 201-205. https://doi.org/10.1109/ICDAR.2009.40

[23] Kessentini, Y., Paquet, T., Hamadou, A.B. (2010). Off-line handwritten word recognition using multistream hidden Markov model. Pattern Recognition Letters, 31(1): 60-70. https://doi.org/10.1016/j.patrec.2009.08.009

[24] Jayech, K., Mahjoub, M.A., Amara, N.E.B. (2016). Arabic handwritten word recognition based on dynamic Bayesian network. The International Arab Journal of Information Technology, 13(6B): 1024-1031.