Electronic Letters on Computer Vision and Image Analysis 21(1):121-129, 2022

Attention-based CNN-ConvLSTM for Handwritten Arabic Word Extraction

Takwa Ben Aïcha Gader[*,°,+] and Afef Kacem Echi[*,−]

[*] *University of Tunis, ENSIT-LaTICE, Tunis, Tunisia*
[°] *University of Sfax, ENIS, Sfax, Tunisia*
[+] *takwa.ben.aichaa@gmail.com*
[−] *afef.kacem@ensit.rnu.tn*

Received 27th of April, 2021; accepted 14th of June 2022

Abstract

The Arabic manuscripts' segmentation into text lines and words is an essential step, where its results directly affect the accuracy of recognition systems. In this work, we tackle the problem of word extraction, a critical and challenging task for many reasons. First, words are usually split into sub-words, and disconnected letters may appear. Second is the presence of ascenders, descenders, and diacritics in Arabic handwriting. Finally, we can cite the skewness problem, which can cause touching characters. This work proposes a deep-learning-based approach for handwritten Arabic word extraction. We adopted an Attention-based CNN-ConvLSTM (Convolutional Long Short-term Memory) model followed by a CTC (Connectionist Temporal Classification) function. Firstly, the text-line input image's essential features are extracted using an Attention-based Convolutional Neural Network (CNN). The extracted features and the text line's transcription are then passed to a ConvLSTM to learn a mapping between them. Finally, we automatically used a CTC to learn the alignment between text-line images and their transcription. We trained the proposed model on a complex KFUPM Handwritten Arabic Text (KHATT) dataset. It consists of complex patterns of handwritten Arabic text lines. The experimental results show a visible efficiency of the used combination on three databases. We ended up with an extraction success rate of 91.7% on the KHATT database, 92.8% on the AHDB database, and a rate of 94.1% on the IFN/ENIT database.

Key Words: Attention mechanism, ConvLSTM, CTC, Deep-learning, Handwritten Arabic word extraction.

1 Introduction

This work deals with word extraction from text lines of unconstrained handwritten Arabic document images. Segmentation into words is considered to be one of the main steps in Optical Character Recognition (OCR), Text Document Analysis (TDA), Pattern Recognition, and Analysis (PRA) systems since word extraction can significantly help the process of recognition. It is still a challenging problem as the handwriting is not constrained, and the writing varies greatly depending on the writer. Moreover, letters are sequentially written in

Correspondence to: takwa.ben.aichaa@gmail.com
Recommended for acceptance by Angel D. Sappa
https://doi.org/10.5565/rev/elcvia.1433
ELCVIA ISSN: 1577-5097
Published by Computer Vision Center / Universitat Autonoma de Barcelona, Barcelona, Spain

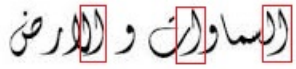

Figure 1: Examples of Handwritten Arabic writing characters that extend to the area of the next characters.

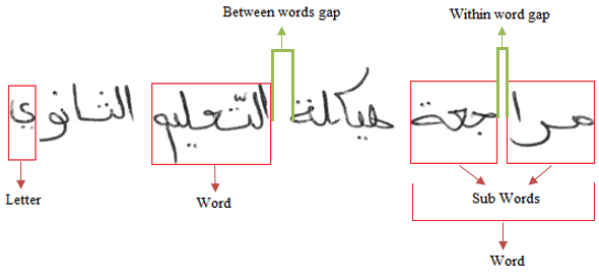

Figure 2: Gap between words and gap within word.

languages other than Arabic, such as Latin and Chinese. These scripts are stringently followed in both document images of handwritten and printed scripts, which differ from Arabic, where words follow an intricate writing style. In Arabic words, before one character is ended, many other ones can be written (for example, see Figure 1). That is due to the space-free writing style, where gaps between characters inside words and those between words do not follow any rules (see Fig. 2). Notice that this is one of the biggest problems in Arabic text segmentation into words.

Recently, several research works have been done in this field. Nevertheless, the most prominent ones are those based on deep learning. They are machine learning approaches that achieve high performance in several classification problems involving character recognition, emotion speech detection, writer identification, car series detection, etc. We were excited to adopt one of the deep learning models to solve the Arabic manuscript text segmentation problem. In this work, we focus on the word extraction problem. In similar work [1], authors used a BLSTM-CTC, a Bidirectional Long Short Term Memory followed by a Connectionist Temporal Classification. The latter performs the mapping between the transcription and the text-line image (see Figure 3 for an example of a handwritten Arabic text-line and its transcription extracted from the KHATT database). They adopted a CNN (Convolutional Neural Network) to extract features to feed the BLSTM-CTC. For ground truth, they used text-line transcriptions, as there was no available labeled ground truth at the word level. Tested on the standard KHATT Arabic database [9], their model achieves an extraction success rate of 80.1%, which is assuredly improvable.

Inspired by the cited work, we propose an end-to-end system using an Attention-based neural network that consists of a CNN-CAtt-ConvLSTM-CTC architecture. It is about a CNN, followed by a spatial attention

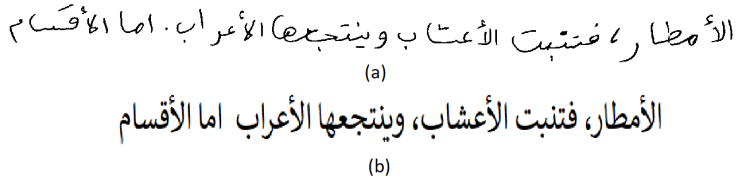

Figure 3: An example of a handwritten Arabic text-line image and its transcription from the KHATT [9] database.

(CAtt), a Convolutional Long Short-term Memory (ConvLSTM), and a CTC, as will be explained later.

The rest of the paper is organized as follows. In section 2, some related works are reported. Section 3 is dedicated to the proposed system for word extraction. In section 4, we discuss the experimental results. We finally conclude and give some prospects in section 5.

2 State-of-the-Art

The first Arabic word extraction research proposed in 2009 by AlKhateeb et al. [2] proposed a component-based approach. They analyzed the connected components based on baselines and reached a correct extraction rate of 85%. In 2010 V. Papavassiliou et al. [3] proposed an SVM-based gap metric for adjacent connected components at the text-line level. They applied a threshold to classify gaps as "within" or "between" words. They tested their method on the ICDAR 2007 Handwriting Segmentation Contest datasets and achieved an F-measure of 93%. A gap metric method was also proposed in 2014 by A. Al-Dmour et al. in [4]. They used a clustering algorithm to identify segmentation thresholds as "within the word" or "between words" gaps. They tested their method on the AHDB dataset [17] and obtained an 84.8% correct extraction rate. Works are, succeeded, and researchers tried to achieve better performances. In 2016, A. Al-Dmour et al. proposed a method based on two spatial measures: connected component length and gaps between them, differentiating the successively connected components in text lines. Connected component lengths are clustered to separate between the groups of letters, sub-words, and words. The clustering is implemented using the Self-Organizing Map (SOM) algorithm [7]. Furthermore, gaps are clustered into two clusters to indicate whether the gap occurs between words or within-a word. The method was tested on the AHDB dataset [17] and achieved a correct extraction rate of 86.3%. Note that most of these methods are based on gap classification. We believe that the spaces between words cannot be used as a relevant characteristic for word extraction due to the different writing styles.

Recently, N. Aouadi et al. [8] proposed an automatic system for Arabic handwritten word extraction and recognition. The system detects segments touching characters, extracts sub-words and words, and recognizes them with a Markovian classifier. They achieved an average recognition rate of 87% on some historical handwritten documents. In 2019, C. Neche et al. [1] used a CNN-BLSTM-CTC neural network architecture for word extraction. They used a CNN to extract features from input handwritten text images to feed the BLSTM network. A CTC follows this latter to perform the mapping between the transcriptions and text-line images. Tested on the KHATT Arabic database [9], an extraction success rate of 80.1% is achieved. In this work, we propose to extend that model using an Attention-based neural network that consists of the CNN-CAtt-ConvLSTM-CTC architecture.

3 Proposed Approach

For word extraction, we have handwritten text-line images as input and text-line transcription as ground truth. The desired output is a sequence of successive extracted words (see Figure 3). To make relations between these three varieties of data, we employed a CNN-CAtt-ConvLSTM-CTC architecture. CNNs are often used in modeling problems related to spatial inputs such as images. They have been proven successful in issues of computer vision, image classification, object detection, etc. LSTMs are utilized to model problems related to sequences and make predictions based on them. They are a special kind of RNN (Recurrent Neural Network), capable of learning long-term dependencies. Note that LSTMs are widely used in NLP (Natural Language Processing) related tasks like machine translation, sentence classification, generation, etc. As standard LSTMs cannot be used directly on spatial input sequences, CNN-LSTMs models are proposed. Their inputs have either a spatial structure (2D structure of images, 1D structure of words in a sentence, paragraph, or document) or a temporal structure (order of pictures in a video or words in a text) or also the generation of output with a temporal structure (words in a textual description). Thus, the CNN-LSTM architecture involves CNN layers

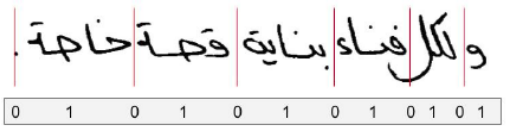

Figure 4: Ground truth provided to the CTC (1: word; 0: space)[1].

for feature extraction on input data combined with LSTM to support sequence prediction. We used a CNN to extract text-line image features in the proposed approach, followed by a spatial attention block to use the inter-dependencies across the feature channels and improve the feature extraction block. The Attention block customizes the weight of relevant channel features according to the correlation of input features. The relevant extracted features are then passed to the ConvLSTM-CTC block. Note that we adopted a ConvLSTM instead of an LSTM. It is a variant of LSTM that contains convolution operations inside the LSTM cells. More precisely, the matrix multiplication is replaced by a convolution operation at each gate in the LSTM cell. By doing so, ConvLSTM captures underlying spatial features by convolution operations in multiple-dimensional data.

Figure 10 gives a global overview of the proposed model. Hereafter more details are given about the retained model's architecture.

(a) **CNN**: CNN is a convolutional neural network commonly used for feature extraction in many deep learning-based domains. A detailed description of the used CNN is offered below:

- **Convolution layers number**: the used CNN is a deep model with five layers; its architecture is presented in table 1 (abbreviations: Max-Pooling (Pool), batch normalization (BN), convolutional layer (Conv)).

Type	**Description**	**Output size**
Input	gray-value text line image	$48 \times 1600 \times 1$
Conv	kernel 3×3	$48 \times 1600 \times 16$
Conv + BN + Pool	kernel 3×3 , pool2 $\times$ 2	$24 \times 800 \times 16$
Conv + BN + Pool	kernel 3×3 , pool 2×2	$12 \times 400 \times 32$
Conv + BN	kernel 3×3	$12 \times 400 \times 64$
Conv + Pool	kernel 3×3 , pool 2×2	$6 \times 200 \times 64$

Table 1: The used CNN architecture.

- **Convolution batch normalization (BN)**: batch normalization is added to the second and third layers between our CNN's convolution and max-pooling operations. The fourth layer is between the convolution and the activation function. We operated a binary batch normalization $(0, 1)$; in other words, each layer, including BN, will standardize inputs with a mean of zero and a standard deviation of one. The momentum parameter is fixed to 0.9.
- **Convolution activation function**: the activation function defines how the input's weighted sum is converted into an output of one node or multiple nodes in a network layer. It controls how well the network model learns the training database in the hidden layers, whereas it defines the model prediction type when appearing in the output layer. The Rectified Linear Unit (ReLU) function has become a well-used activation function. It is a reliable function and accelerates the convergence by six times compared to tanh and sigmoid. It calculates the function $f(K) = \max(0, K)$. In other

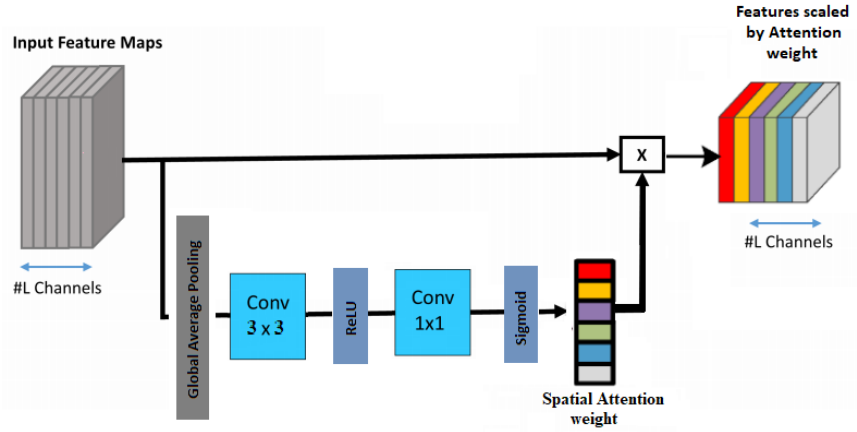

Figure 5: Spatial Attention block.

words, it is just a threshold at zero. The ReLU activation function was used in all the layers in the proposed CNN since the last one is not making predictions.

(b) **The Attention block**:CNN is a visual feature extractor for most deep learning-based methods. Its convolution kernel maintains the filtering function of feature detection. The features of each channel define the weights of the image on different convolution kernels. To improve the CNN feature extraction process, we can follow it with an attention block. Generally, attention directs your focus on something; it is the mental process of selectively focusing on one or some things while neglecting others. In deep neural networks, the attention mechanism likewise tries to implement the same idea of selectively concentrating on a few essential things while ignoring others. An Attention block following a CNN may skillfully adjust the relevant channel features' weight depending on the input feature correlation. More precisely, it improves the network's capacity for feature representation by modeling the habits of each feature map's channel. It first applies a global pooling in the channel dimension to extract each channel's global information. Afterward, it models the correlation between channels adaptively. And it uses this correlation to weight each channel to achieve the objective of feature response and re-calibration.

- **The used spatial attention block**: Unlike channel attention, which concentrates on what is essential in an input image, as done in the object detection problem, we use spatial attention as a supplement to learning where to direct attention for word extraction problems. The used spatial attention module is displayed in Figure 5. First, the CNN model provides the channel-wise refined feature map. Then an average pooling operation is applied to the channel dimension to perform an efficient feature descriptor and highlight informative regions. Then we operated a 3×3 convolution to extract multi-scale features. Afterward, we applied a 1×1 convolution to decrease the channels. Therefore, the input and the output of our spatial attention blocks have the same channel number. Then we used the Sigmoïd function to get the spatial attention weights. The resulting spatial attention weights and the input image features are fused for the feature attention calculation.

(c) **ConvLSTM**: Remember that LSTM is a variant of RNNs, and all RNNs try to track some evolving state over time. LSTM is rated as one of the prevailing methods due to its memory cell, which can accumulate and neglect states being tracked from one-time step to another. This innovation allows the LSTM to alleviate one of the main limitations of other RNNs by ignoring some states before becoming used. The memory cell in the LSTM enables the model to choose which states will be suitable to remember and which ones will be securely forgotten. LSTMs also grant protection against vanishing gradient problems (gradients of the loss function that turn into zeros) as they propagate backward within the network while

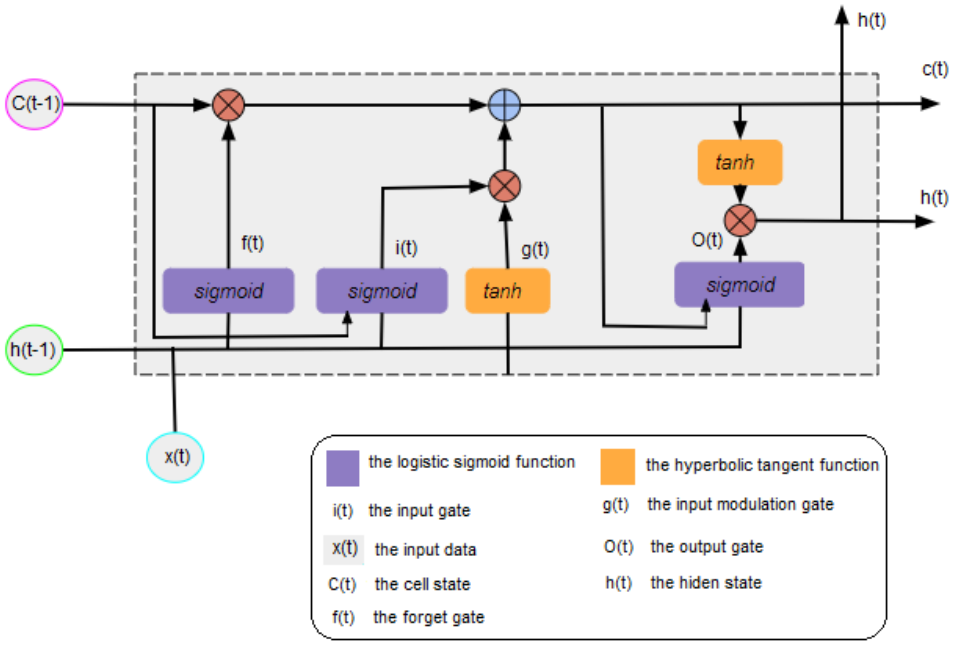

Figure 6: Schematic representation of the ConvLSTM cell.

training. It will be beneficial to retain relevant spatial information to improve the performance of the word extraction problem. We used ConvLSTM instead of the LSTM to preserve the spatial structure of the handwritten text image, as ConvLSTM can catch the spatial information better than the LSTM. That is performed by operating a convolution operation just like CNN does. The ConvLSTM model and the LSTM one have the same core, where the previous layer's output serves as input for the next one. The principal difference is that the ConvLSTM can obtain temporal and spatial relations. The strength point of using the ConvLSTM is that convolutional operations influence spatial attention as the word regions generally include a more activating solid response. The ConvLSTM unit has inputs, gates, outputs, and memory cells (see Figure 6). Each unit makes a decision based on the current input, the previous unit's output, and the last unit's memory. It then creates a new output and modifies its memory. The current block forms a new output and adjusts its memory by making a decision.

The following Equations define the gates and outputs of the ConvLSTM cell:

$$wi_t = sigmoid(W_{ix} \oplus x_t + W_{ih} \oplus h_{t-1} + W_{ix} \odot c_{t-1} + b_i) \tag{1}$$

$$f_t = sigmoid(W_{fx} \oplus x_t + W_{fh} \oplus h_{t-1} + W_{fx} \odot c_{t-1} + b_f) \tag{2}$$

$$o_t = sigmoid(W_{ox} \oplus x_t + W_{oh} \oplus h_{t-1} + W_{ox} \odot c_{t-1} + b_o) \tag{3}$$

$$g_t = \tanh(W_{gx} \oplus x_t + W_{gh} \oplus h_{t-1} + b_g) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{5}$$

$$h_t = o_t \odot \tanh(c_t); \tag{6}$$

Where:

– $x_t \in R^d$: is the current input, $t \in \{1..T\}$ is the observations sequence, T is the observations maximum length and d is the input depth.

- $h_{t-1} \in R^{h \times w \times r}$: is of the previous ConvLSTM block output, where h and w are images height and width and r is the recurrent cells number.
- $C_{t-1} \in R^{h \times w \times r}$: is the previous ConvLSTM block cell memory's output.
- $h_t \in R^{h \times w \times r}$: is the output of the current block.
- $C_t \in R^{h \times w \times r}$: is the current cell memory output.
  $\Rightarrow$ $\mathrm{x}_t, c_t, h_t, i_t, f_t$, and o_t are three-dimensional tensors; the first one defines temporal information and the second and third ones define the rows and columns of spatial information.
- $W \in R^{(r+d) \times r}$: are recurrent weights adapted while training. process
- b: are the bias terms.
- i_t :is the input gate.
- f_t: is the forget gate.
- O_t: is the output gate.
- g_t: is the candidate gate.
- $\oplus$: is the 2D spatial convolution operation.
- $\odot$: is the Element multiplication (pixel-wise product).
- $Sigmoid$: is the sigmoid function mapped real numbers to (0,1).
- Tan : is the tan function mapped real numbers to (-1,1).

(d) **CTC**: It was first proposed by Graves et al. [11] to address the problem of alignment in speech recognition. It is a function that enables artificial neural networks to identify labels for sequenced temporal data. The CTC function is required for the word extraction problem because we have no prior alignment between the ConvLSTM's output and the ground truth fed to the CNN. Our general system used an attention-CNN to extract a sequence of features and a ConvLSTM to propagate information via this sequence. It returns a matrix of class (1 for word and 0 for space between words) scores for each sequence element. This matrix is used for training and prediction. The CTC block uses this matrix for the neural network training by loss calculation, and for inference, it decodes the matrix to obtain the sequence of words-spaces included in the input image. It is used to overcome the unknowing alignment problem between the input and the output. The CTC Loss function guides the training of the proposed neural network. The features matrix resulting from the ConvLSTM and the ground truth (see Figure 4) are given to the CTC loss function. This latter attempts all possible alignments of the ground truth sequence in the image and calculates all alignment scores. A high sum value implies a high GT sequence score. Hereafter are details of the CTC algorithm.

Algorithm

The CTC function may calculate a probability for any Y based on a given input X. It attempts to find an alignment between inputs and outputs. We first provide a way of extracting these alignments and then, based on them, we provide the methods of calculating the loss function and making predictions.

- **Alignment (encoding)**: CTC does not need input and output alignment. It just wants to calculate the probability of an output Y given an input X. It operates by computing the probabilities of all possible ways of alignment between Y and X. The total probability of a Y is calculated by summing the probabilities of its different alignments. One or more input elements can align a single output element, and the opposite is false. The length of each output Y cannot be larger than the input length. The blank class "$-$" is added to the target label vocabulary for two reasons. Firstly, it separates the duplicated label in the label sequence, and secondly, it is used as the default label

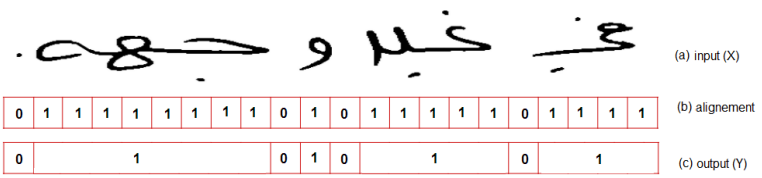

Figure 7: An example of an image alignment.

for unlabeled data. It is noticed that a single word can take multiple horizontal positions. One of the CTC's strong points is that we do not have to tell him the position or width of the words in the image. The Softmax function regularizes the outputs to get the X to Y distribution at each time step.

Figure 7 is an example of annotation for each horizontal position of a given image. Where (a) is an input image X having a horizontal position length of 22 and (c) is the output $Y = [0, 1, 0, 1, 0, 1, 0, 1]$. CTC assigns an output class ("1", "0", or "$-$") to each input step and collapses repeats to align X and Y. The input and the alignments resulting from the CTC have the same lengths. To map the alignments (b) to the Y output (c), we merge the repeats, where each successive time step has the same classes merged, and remove the blank label "$-$".

– **Loss calculation**: we should calculate the training samples' loss value for each pair of images and GT sequences to train the neural network. This latter outputs a matrix including a class score for each time step. All possible alignments (or paths) of the GT sequence scores are summed to compute the loss value and calculate the score of a single alignment, and all-time step class scores are multiplied (see Figure 8). Following are the relative formulas:

- The probability of a single path π can be computed as:

$$p(\pi|X) = \prod_{t=1}^{T} p(o^t_{\pi^t}|x^t); \tag{7}$$

Where, π^T is the time step t label for the path , O^t is the model output at the time step t and x is the input at time step t. Figure 8 shows an example of a resulted matrix with $4 - time$ steps and two classes ("1" for a word and "0" for a space inter-words). The black lines present the possible path representing the input image. Its score is equal to $0.4 \times 0.5 \times 0.6 \times 0.7 \times 0.9 = 0,076$.

- Our goal is to train our model to maximize the probability given to the correct sequence (or answer) for a provided input. So, we will need to calculate the conditional probability effectively, our loss function. It is calculated by a sum of the probabilities of all possible CTC paths; its formula is presented as follows:

$$p(Y|X) = \sum_{\pi \in F^{-1}(Y)} p(\pi|X); \tag{8}$$

Where F is the mapping function, performing paths mapping to the label sequence. It first merges the identical successive labels into one and then removes the blanks.

- To train our proposed neural network, we aim to minimize the following objective function;

$$\sum_{(X,Y) \in D} -\log p(Y|X); \tag{9}$$

Where D is the training dataset.

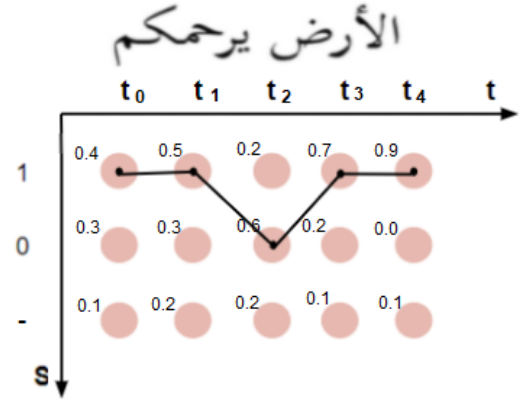

Figure 8: Neural network output's matrix. The class probability is written next to each matrix entry. The black lines present the path representing the input image.

– **Inference**: Now to use the trained neural network for inference; to estimate the most probable sequence given the network output matrix. To do so, we used the best path-decoding algorithm. Which first estimates the best path by assuming the most label ("1", "0" or "−") per time-step, and then it removes from the path duplicated labels, then all blanks. The rest denotes the predicted sequence. We proposed a post-treatment to associate the predicted sequence with the input image. Notice that the input image and the labeled sequence have the same length, so we project every predicted sequence's 0 labels on the input image by a vertical red line example (see Figure 9 for an example).

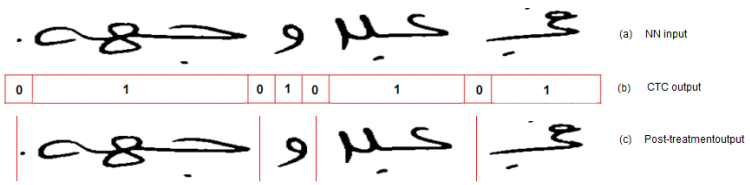

Figure 9: An example of the post-processing process.

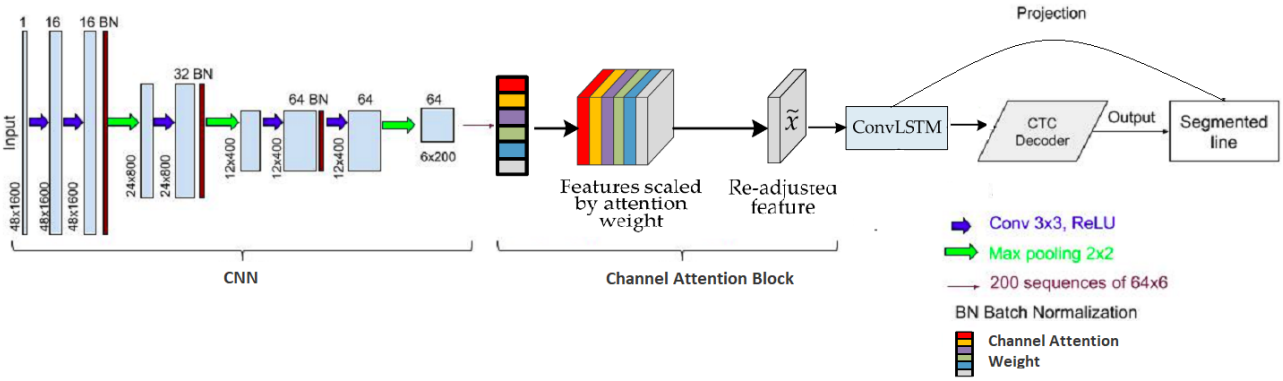

Figure 10: The proposed word extraction model.

4 Experimental Results

4.1 Used Databases

To assess the proposed Attention-based CNN-ConvLSTM model's performance and compare it to those of some related works, we used three databases: KHATT, AHDB, and EFN/ENIT, where the KHATT database involves more complex writings than IFN/ENIT and AHDB databases.

- **KHATT** [9] (KFUPM Handwritten Arabic TexT): is a database of unconstrained handwritten Arabic texts written by 1000 different writers. The database covers 2000 unique paragraph images and their segmented line images (source text from various topics like arts, education, health, nature, and technology), 2000 paragraph images containing similar text, each covering all Arabic characters and shapes and their segmented line images, and a free paragraph written by writers on any topic of their choice. The images are offered with manually verified ground-truth (Unicode Transcription, see Figure 3) and their Latin representation.
- **IFN/ENIT** [18]: Created in 2002 by the Institute for Communications Technology (IfN) at Technical University Braunschweig, Germany, and the Ecole Nationale d'Ingenieurs de Tunis (ENIT), Tunisia, contains handwritten Tunisian town. It contains material for training and testing Arabic handwriting recognition software. More than 2200 binary images of handwriting sample forms from 411 writers; about 26,000 binary word images have been isolated from the forms and saved individually for easy access. A ground truth file for each word in the database has been compiled. This file contains information about the word, such as the position of the word's baseline and information on the individual characters used in the word.
- **AHDB** [17]: it includes words used to write legal amounts on Arabic checks and free writing pages of 100 writers. The database contains 105 forms.

4.2 Train

We trained the model for 100 epochs. The training settings are represented in Table 2. Figure 11 shows the training curves, which reflect a good fit, and Figure 14 displays an example of the proposed system's result. The ConvLSTM gives 200 probabilities for the class space (0) and 200 probabilities for the class word (1).

Image pre-processing:	We used 4800 line images for training (where: 80% are taken for the training and 20% for the validation) and 200 for testing. Images are resized to 48×1600. No further processing.
Training settings:	Number of epochs:100. Initial learning rate: 10^{-6}. Weights initializer: Xavier. Optimizer: Adam. Batch-size per epoch: 64. Evaluation metrics: Loss and F-measure

Table 2: Parameters settings of the proposed architecture.

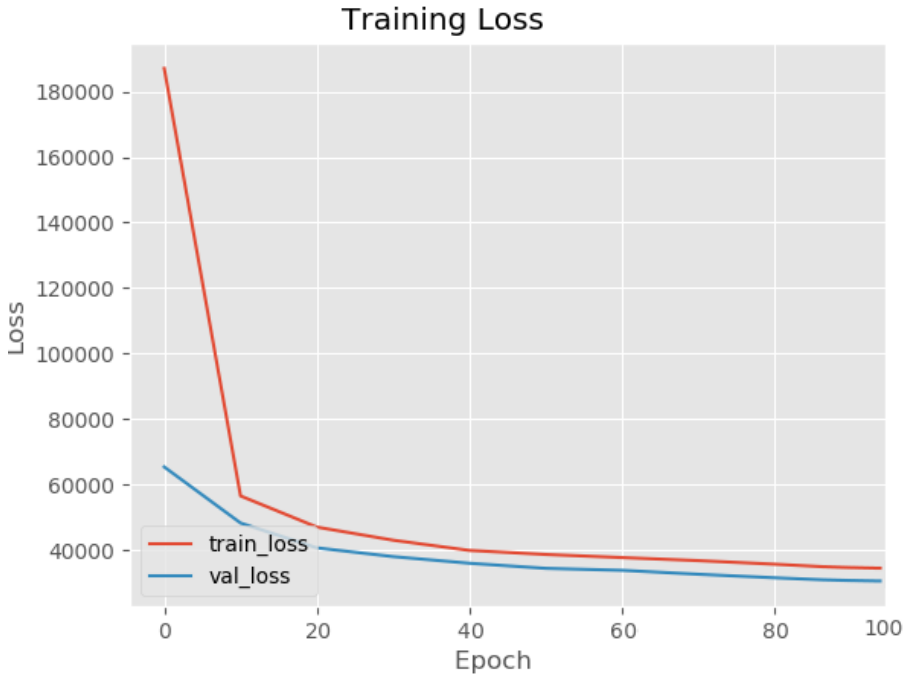

Figure 11: Training and Validation Loss curves.

4.3 Test

We used the F-measure metric on the test database to assess the performance of our proposed system. Its formula is presented as follows:

$$F - Measure = \frac{(2 \times Precision \times Recall)}{Precision + Recall} = \frac{tp}{tp + \frac{1}{2} \times (fp + fn)}; \tag{10}$$

Where the formulas of Precision and Recall are presented in equations 11 and 12 and tp is the number of true positives, fp is the false positives and fn is the number of false negatives on the predicted sequences on the test dataset (see Figure 12).

$$Precision = \frac{tp}{tp + fp} \tag{11}$$

$$Recall = \frac{tp}{tp + fn} \tag{12}$$

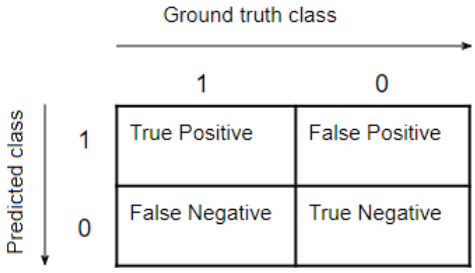

Figure 12: Possible prediction classes in binary classification.

The predicted output is either ”0” or ”1” in binary classification. For the testing dataset, we called a true positive if the predicted class is ”1” and the ground truth class is also ”1”. And if the ground truth class is ”0” and the predicted one is ”1”, that is a false positive. While if the ground truth class is ”1” while the predicted one is a ”0”, it is named a false negative. And finally, if both the ground truth class and the predicted one are ”0”s, that is a true negative. Table 3 shows the F-measure values, of our system, of the original paper of [1] and of other works. As it can be seen, our proposed method outperforms the mentioned state-of-the-art methods. This improvement is due to the two main extensions made to the original model. First, the ConvLSTM model applies convolutions to the output of previous blocs, enabling each succeeding ConvLSTM block to identify more complex and abstract relations to the provided data. Second, the used spatial attention mechanism improves feature extraction. Figure 14 presents an association of the input image and an alpha channel created from its CTC probabilities of the class space. We resized the input image to the 6×200 shape before associating the alpha channel to the input image to have the same shape with the probabilities vector. Then we resized the obtained image to the original shape:64×1800, presenting the segmented line after the probabilities projection. And Figure 14 displays an example of the final CTC classification (after eliminating duplicated labels and all blanks from the assigned alignment) projected on the input image by red vertical lines.

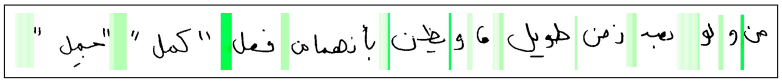

Figure 13: Association of the input image and the alpha channel created from the probabilities of the class space.

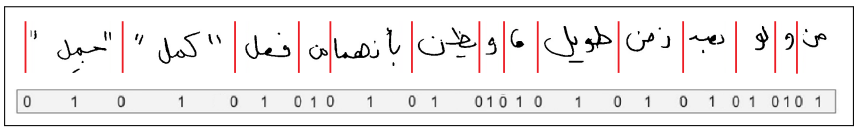

Figure 14: A result of the proposed system

5 Conclusion and Future Work

In this work, we proposed an efficient method to solve the Arabic handwritten word extraction problem based on the work presented in [1]. In the original work, the authors used a CNN with a BLSTM followed by a CTC to find an alignment between the text-line transcription and the text-line image. We replaced the BLSTM with a ConvLSTM since it achieves visible success in several image processing fields. Otherwise, we used

Architecture	**Method**	**Dataset**	**Rate (%)**
[12]	Calculation of within-word and between-words gaps (Bayesian criteria)	IFN/ENIT	85.0%
[16]	Metric-based segmentation	IFN/ENIT	72.45%
	ESL-based segmentation	IFN/ENIT	87.9%
	ESL + Metric- based segmentation	IFN/ENIT	93.135%
[13]	Classification of gaps to "within word" and "between words" using a threshold identified by an enhanced gap metric	AHDB	84.3 %
[14]	Classification of interconnected components spaces into inter-and intra-words using a threshold calculated for each text line	AHDB	87.9 %
[15]	Classification of within-word and between-words gaps (Kmeans clustering)	AHDB	84.8 %
[5]	Classification of within-word and between-word gaps and connected component lengths (SOM clustering: Kohonen Neural Network)	AHDB	86.3%
[1]	Deep architecture: CNN+BLSTM+CTC	KHATT	80.1%
Proposed method	Deep architecture: CNN+Attention+ConvLSTM+CTC	AHDB	**92.8**%
		KHATT	**91.7**%
		IFN/ENIT	**94.1** %

Table 3: comparing results from different models.

an Attention mechanism on the bottom of the CNN; the latest can adaptively weigh the channels of feature maps resulting from the CNN to select more powerful features. These modifications allow the improvement of the model's results. It is a promising result compared to the original paper's one and state-of-the-art methods. We plan to improve the word extraction performance using a specific post-treatment to correct the wrong segmentation in future work. We also plan to train the model on other databases and deal with various kinds of unconstrained writing and alphabets.

References

[1] C. Neche, A. Belaïd and A. Kacem-Echi: Arabic Handwritten Documents Segmentation into Text-lines and Words using Deep Learning, ASAR, 2019.

[2] J.H. AlKhateeb, J. Jiang, J. Ren and S. Ipson S: Interactive knowledge discovery for baseline estimation and word segmentation in handwritten Arabic text, Strangio MA (ed), Recent advances in technologies, 2009.

[3] V. Papavassiliou, Th. Stafylakis, V. Katsouros and G. Carayannis: Handwritten document image segmentation into text lines and words, Journal Pattern Recognition, V. 43, N. 1, pp. 369-377, 2010.

[4] A. Al-Dmour and F. Fraij: Segmenting Arabic Handwritten Documents into Text lines and Words, International Journal of Advancements in Computing Technology , V. 6, N. 3, pp. 109-119, 2014.

[5] A. Al-Dmour and R.r: Word Extraction from Arabic Handwritten Documents Based on Statistical Measures, International Review on Computers and Software (I.RE.CO.S.), V. 11, N. 5, pp. 1828-6003, 2016.

[6] S. Al-Ma'adeed, D. Elliman and C.A. Higgins: A data base for Arabic handwritten text recognition research, Proceedings of Eighth International Workshop on Frontiers in Handwriting Recognition, pp. 485–489, 2002.

[7] T. Kohonen: The self-organizing map, Proceedings of the IEEE, (78):9, pp. 1464-1480, 1990.

[8] N. Aouadi and A. Kacem Echi: Word Extraction and Recognition in Arabic Handwritten Text, International Journal of Computing & Information Sciences Vol 12, N. 1, pp. 17 – 23, 2016.

[9] A. M. Sabri, A. Irfan, G. Wasfi, M. Al-Khatib, M. Tanvir Parvez, V. Märgner, A. Fink Gernot: KHATT: An open Arabic offline handwritten text database, Pattern Recognition, V. 47, N. 3, pp. 1096-111, 2014.

[10] Li Y, Xu H, Bian M, Xiao J.: Attention Based CNN-ConvLSTM for Pedestrian Attribute Recognition, Sensors (Basel),V. 20, N. 3, 2020.

[11] A. Graves, S. Fernandez, F. Gomez and J. Schmidhuber: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, International Conference on Machine learning, pp. 369-376, 2006.

[12] J. H AlKhateeb, J. Jiang, J. Ren and Stan S Ipson : Component-based Segmentation of Words from Handwritten Arabic Text, International Journal of Computer and Information Engineering, (2):5, pp. 1428-1432, 2008.

[13] A. Al-Dmour, F. Fraij : Segmenting Arabic handwritten documents into text lines and words, Int. J. Adv. Comput. Technol, (IJACT), 6(3), 2014.

[14] A. Lamsaf, A. Mounir, B. Siham and F. Youssef: Text Line and Word Extraction of Arabic Handwritten Documents, Innovations in Smart Cities Applications, 2, 2019.

[15] A. Al-Dmour and F. Fraij: Segmenting Arabic Handwritten Documents into Text lines and Words, International Journal of Advancements in Computing Technology 6(3), pp. 109-119, 2014.

[16] Amani T. Jamal, Nicola Nobile, and Ching Y. Suen: End-Shape Recognition for Arabic Handwritten Text Segmentation, Artificial Neural Networks in Pattern Recognition, pp. 228-239, 2014.

[17] S. Al-Ma'adeed and D. Elliman and C. A. Higgins: A data base for Arabic handwritten text recognition research, Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition, pp. 485-489, 2002.

[18] M. Pechwitz, S. S. Maddouri, V. Märgner, N. El-louze, and H. Amiri: Ifn/enit - database of handwritten arabic words, Proc. of CIFED 2002, pp. 129–136, 2002.