Deep Learning and Recurrent Connectionist-based Approaches for Arabic Text Recognition in Videos

Sonia Yousfi
Orange Labs – France Telecom
35510 Cesson-Sévigné, France
Email: sonia.yousfi@orange.com

Sid-Ahmed Berrani
Orange Labs – France Telecom
35510 Cesson-Sévigné, France
Email: sidahmed.berrani@orange.com

Christophe Garcia
University of Lyon, INSA-Lyon
LIRIS, UMR5205 CNRS
69621 Villeurbanne, France
Email: christophe.garcia@liris.cnrs.fr

Abstract—This paper focuses on recognizing Arabic embedded text in videos. The proposed methods proceed without applying any prior pre-processing operations or character segmentation. Difficulties related to the video or text properties are faced using a learned robust representation of the input text image. This is performed using Convolutional Neural Networks and Deep Auto-Encoders. Features are computed using a multi-scale sliding window scheme. A connectionist recurrent approach is then used. It is trained to predict correct transcriptions of the input image from the associated sequence of features. Proposed methods are extensively evaluated on a large video database recorded from several Arabic TV channels.

I. INTRODUCTION

Given the huge available amount of videos via TV Channels and recording devices, automatically indexing and searching such type of documents is an issue of great importance for many applications. Embedded text in videos is one of the most relevant sources of high-level semantic clues that are used for this purpose. Research studies regarding Optical Character Recognition (OCR) have been widely developed during the past five decades for languages like English and Chinese addressing specially scanned documents. However, human history is being also recorded in other forms (videos and images), with many other languages.

In this context, our work focuses on Arabic embedded text recognition in videos. Arabic is used by more than half of a billion people in the world. Many Arabic big news channels, compared to BBC and CNN, appeared in the past two decades. Proposing OCR solutions for this use-case is, thus, very useful for many applications. However, Arabic script is very challenging to recognize. It is semi-cursive. The alphabet of Arabic language contains basically 28 characters whose shapes change according to the position in the word (isolated, beginning, middle or end) [1]. One character can have from 2 to 4 different shapes as shown in Figure 1.a. Additional difficulties come from the presence of points and diacritics like the ‘Hamza’ (cf. Figure 1.b), and the high similarity between some characters that can hardly be differentiated without context (cf. Figure 1.c). Other challenges are related to text variations (size, font, color...) and video acquisition conditions (complex background, luminosity variations, low resolution...).

The first step of Arabic text detection within video frames has been already tackled in our previous work [2]. In this paper, we focus on the recognition step. Our methodology consists in a multi-scale scanning scheme of the input text image for features sequence generation without explicit character segmentation. Text transcription is considered as a character sequence labeling problem. It is performed using a recurrent connectionist approach. A Recurrent Neural Network (RNN) coupled with a Connectionist Temporal Classification component (CTC) is trained to perform the sequence labeling without any prior segmentation information. The RNN-CTC scheme strongly depends on what it receives as features. We propose in this work not to use hand-crafted features. These features are biased by the knowledge of the human expert and do not cope very well with complex and noisy input text images. Instead, we propose to learn features using deep learning models namely deep auto-encoders and Convolution Neural Network (ConvNet). Each model is applied separately within the scanning scheme of the input text image leading to different OCR solutions; each one feeds the RNN-CTC scheme with different learned features.

The rest of the paper is organized as follows. Section II gives an overview of related works. Section III describes the proposed solutions. Experiments and results are then presented and discussed in Section IV. Section V concludes the paper.

Fig. 1: Some Arabic script specificities.

(a) Hamza and different letters

(b) Rows and column

(c) 4 different letters

1In the rest of the paper, ‘text image’ refers to the text region that is detected and extracted from a video frame.
Although it is intuitive, segmentation-based recognition is an error-prone specially for cursive script with overlapping and ligature cases [7] as well as for texts with complex background. Other segmentation-free OCR systems have been proposed [8]. Some of these approaches tend to sequentially analyze text line features generated using most often a sliding window. Based on a character model, each observed subset of features is probabilistically classified as a specific character without any prior knowledge about character boundaries. In [9], this technique is used and a set of 16 features, based on the sum of black pixels per strip, is extracted for each window. The resulting feature vectors are then fed to a Hidden Markov Model (HMM) for character sequence labeling. In [10], sub-character HMM models have been considered in order to take into account common patterns between different characters as well as between different shapes of the same character.

Under the same paradigm, other approaches use RNNs instead of HMMs [11]. RNNs, as discriminative models, focus entirely on searching for the correct labels unlike HMMs which are generative models. They are also able to model non-linear inter-dependencies between input features. In practice, HMMs generally use mixture of diagonal Gaussian to model the distribution of input features, which makes it limited to decorrelated or locally inter-dependent inputs. Another advantage of RNNs is their ability to take into account the context to estimate the probability of each observation. This point has received a particular attention in Bidirectional Long-Short Term Memory-based RNNs (BLSTM). Long range inter-dependencies, in both past and future directions, are considered for sequence labeling. Combined with the Connectionist Temporal Classification component, the BLSTM can learn text transcription using un-segmented data. This BLSTM-CTC scheme has shown superior performance over HMMs in Arabic and Latin handwriting recognition [12,13]. It has also been successfully used for Latin video OCR [14].

Other segmentation-free approaches have been recently proposed. They show good results for Latin scene word recognition. They rely on a holistic paradigm of word images. In [15], a ConvNet is trained to classify large set of word images across a large dictionary. In [16], both images and labels are projected to the same subspace, learned using a structural SVM, in order to achieve the best matching.

III. THE PROPOSED METHODS

The proposed recognition scheme takes as input a text image without any pre-processing operations or prior segmentation. In a first step, the input text image is transformed into a sequence of relevant features. The objective is to emphasize the relevant information at different levels (character, sub-characters, words, sub words...). In a second step, these features are sequentially analyzed and used in order to deduce the right transcription (cf. Figure 2). The first step relies on a combination of a multi-scale window-based scanning scheme and deep neural models in order to represent the text image by a sequence of learned feature vectors. In this paper, we propose three different ways to perform feature extraction: the two first approaches are based on deep auto-encoders while the third one is a based on convolutional neural network. The second step of feature sequence labeling is based on a BLSTM-CTC network.

A. Text feature extraction

Text images vary in font and scale. Even in the same text image, Arabic characters have different shapes with huge variation in height and width. To cope with that, we use sliding windows in a multi-scale scanning scheme of the input text image to cover the different possible positions of characters. As shown in Figure 3, for an input text image of height \( h \), we apply the scanning procedure by a step of \( \frac{h}{4} \) (empirically set-up). It is equal to the median of thinnest character width. A set of 4 windows is applied at each time-step (scanning position), having the same height \( h \) and different experimentally fixed widths, namely \( \frac{h}{4}, \frac{h}{2}, \frac{3h}{4} \) and \( h \).

![Fig. 3: Multi-scale scanning procedure.](image)

Each of these 4 windows is a local view of the text image at a given position and scale. Our goal in this stage is to find a good representation that captures relevant text structures for the next step of recognition. The core idea is to deeply learn a function that captures input data regularities while being robust to noise, font, scale and background variations.

In this paper, we explore three main deep learning models: Deep Belief Networks [17], Multi-Layer Perceptron as deep auto-encoders [18] and Convolutional Neural Networks (ConvNet) [19]. The first two models learn character reconstruction whereas the ConvNet learns an adequate representation for character classification. The three models are trained using a set of gray scale character images belonging to 80 classes\(^2\): 62 letter shapes including the ‘Space’, 10 digits and 8 punctuation signs used in Arabic TV broadcast (‘أ’, ‘ب’, ‘ج’, ‘د’, ‘ظ’, ‘غ’, ‘،’, ‘؟’, ‘،’). As will be explained in the following subsections, labels of characters are needed only by the ConvNet. For the other two models, the training procedure is fully unsupervised.

\(^2\)We are interested here in feature extraction and not in classification. All of the letter shapes have not been taken into account. Very similar letter shapes have been grouped into a unique class. However, for recognition (Section III-B1), we take into account 130 classes including different shapes of the 28 letters.
1) Deep Belief Networks:

A Deep Belief Network (DBN) is a feed-forward neural network with one or more layers containing hidden units often called feature detectors. A particularity of DBN is that the learning procedure of generative weights can be layer-wise. The values of latent variables of a pair of layers can be learned at a time. This is done under the assumption that the internal states of one layer are the input data of the other one. In this work, we use an instance of deep learning strategy: An unsupervised pre-training of Restricted Boltzman Machines (RBM) followed by a fine-tuning procedure in a supervised manner. RBM can be seen as a bidirectional bipartite graph of stochastic visible and hidden units whose joint probability is defined as: \( p(v, h) = \frac{1}{Z} \exp(-E(v, h)) \), where \( v \) and \( h \) are respectively the visible and latent variables, \( E \) is an energy function and \( Z \) stands for the partition function.

In this work, four RBMs with \([2000 \times 1000 - 500 \times 100] \) units are pre-trained using the contrastive divergence algorithm. We clamp a \( 28 \times 28 \) character input image, with normalized real values, directly on the 2000 Gaussian units of the first layer. The next hidden layers with 1000, 500 and 100 binary units achieve a dimensionality reduction and explore high-level data structures. The RBMs are then ‘unrolled’ to produce deep encoder and decoder as shown in Figure 4.a. The resulting auto-encoder is then fine-tuned using the back-propagation of error derivatives to find the optimal character reconstruction. The learned model is separately applied on each scanning window. Each window is then encoded by a feature vector of 100 values, corresponding to the outputs of the last layer of the encoder (cf. Figure 4.a).

2) Auto-Encoder based on Multi-Layer Perceptron:

An auto-encoder can be simply built using a Multi-Layer Perceptron (MLP). Given an input character image, a feed-forward network learns to produce an output that is identical to the input. In our work, as shown in Figure 4.b, a 3-layered neural network with \([120 \times 100 \times 120] \) fully connected hidden units is used. Employing more than one hidden layer with non-linear units in the auto-encoder enhance the ability to capture multi-modal features. The network is then trained using the back-propagation algorithm. Like DBN, the resulting encoder is then applied to map each normalized scanning window in the new learned space with a code of 100 feature values (cf. Figure 4.b).

Here, the 2\textsuperscript{nd} layer of the encoder is the feature layer.

ConvNet is a bio-inspired hierarchical network that has been successfully used for pattern classification [19]. It is able to learn jointly feature extraction and classification using a pipeline of convolutions, sub-sampling and neural layers.

ConvNet is based on three main hierarchical aspects namely local receptive fields, weight sharing and spatial sub-sampling.

In this work, a ConvNet with 6 layers is proposed for character classification (cf. Figure 5). The first layer \( C_1 \) contains \( n_{C1} = 6 \) convolution maps of the input with \( 5 \times 5 \) different trainable masks. This ensures a first level feature extraction (end points, corners...). The 2\textsuperscript{nd} layer \( S_1 \), with \( n_{S1} = n_{C1} \) feature maps, is a sub-sampling layer of previous maps. This reduces their sensitivity to shifts, distortions and variations in scale and rotation. The 3\textsuperscript{rd} layer \( C_2 \) applies both feature combination and a \( 3 \times 3 \) convolution on \( S_1 \) with \( n_{C2} = (n_{S1}\times2) + n_{S1} = 27 \) feature maps. The 4\textsuperscript{th} layer \( S_2 \) is similar to \( S_1 \) with 27 maps of \( 7 \times 7 \) pixels. These maps are then fully connected to a MLP for feature classification with one hidden layer \( N_1 \). This layer contains 100 neurons connected to a softmax layer \( N_2 \) with 81 classification neurons (the 80 character classes and an additional ‘Rubish’ class).

Once learned, the ConvNet is used to compute features for each normalized scanning window. The feature vector corresponds here to the 100 output activations of \( N_1 \).

![Fig. 4: Architectures of auto-encoders.](image)

B. Text recognition

At this level, each text image is a sequence \( X \) of length \( T \). At each time-step \( t \) (scanning position), each of the 4 windows is presented by 100-dimensional vector of learned features. Thus, each element of the sequence \( X \) is a 400-dimensional vector. The text recognition task is seen as a temporal classification problem. The core of our solution is a Bidirectional Long-Short Term Memory network (BLSTM) coupled with a CTC component to learn feature sequence labeling without pre-segmented data (cf. Figure 2). The BLSTM is trained to produce, for an input sequence \( X \), a sequence of character probabilities \( \hat{y}_t = p(l_t|X) \), where \( l_t \) is a character class.

1) BLSTM:

The LSTM is a particular architecture of recurrent neural networks. It has been proposed to cope with long-range dependencies in sequential data using a special memory cell with three multiplicative gates to control information access. Bidirectional LSTM network [12] consists of two LSTM networks. The first handles the input sequence in the forward direction and the other in the backward direction. This architecture allows the BLSTM to take into account past and future contexts. In this work, the network takes as input a sequence of at maximum \( T = 200 \) feature vectors. Each vector is of dimension 400 whose values are normalized between \(-1 \) and \(1\). The input is fully connected to the forward and backward LSTM hidden layers. Both of these layers are connected to a third non-recurrent layer. The output layer is a softmax function that produces predicted character probabilities for each time-step and character. At this stage, each letter at a specific location in a word is considered as a class (label). Hence, we have

![Fig. 5: ConvNet architecture.](image)
130 classes (including letters, digits and punctuations) and an additional class ‘Blank’ for no character case. The network is trained using the Back-Propagation Through Time algorithm (BBTT).

2) Connectionist Temporal Classification (CTC):

During the training phase of the BLSTM, the CTC [12] is used in order to estimate the error vector to back-propagate at each time-step using unsegmented data. For an input, only the target sequence of characters is needed. For this, CTC inserts the additional class ‘Blank’ between each two labels of the target transcription. The goal of the CTC is to get a BLSTM sequence output with peaks corresponding to the target labels and in the same order. Therefore, during training, the CTC tries to minimize an objective function using a process inspired by HMM backward-forward algorithm [20]. Once trained, a CTC decoding scheme is applied on the BLSTM softmax outputs using the best path decoding algorithm. After removing all the ‘Blanks’, the most likely sequence of labels the input is hence deduced.

IV. EXPERIMENTS

Datasets and evaluation measures

A total of 47 hours of videos have been recorded from Arabic TV news channels and manually annotated. A subset of 17 hours, called ArabCharSet, has been used to train feature extraction modules. It contains 46,689 images of single letters and punctuations. The remaining 30 hours have been used to generate text images. Two subsets, ArabTrainText (7000 text images) and ArabValidText (673 text images), have been used respectively for the training and the validation of BLSTM. Another dataset, ArabTestText (900 text images) has been used for testing. Figure 6 illustrates examples from these datasets.

(a) examples of character images

(b) examples of text images

![Fig. 6: Examples of character and text images.](image)

The results have been evaluated using the following measures: the character recognition rate (CRR), the word recognition rate (WRR), the word recognition rate including words with at most one wrongly recognized character (WRR-1C) and the whole text recognition rate (TRR). Given the recognized text (RT) and the ground-truth (GT), these measures are computed as follows:

\[
\text{CRR} = \frac{\#\text{characters} - \sum \text{EditDistance}(\text{RT}, \text{GT})}{\#\text{characters}}
\]

\[
\text{WRR} = \frac{\#\text{words correctly recognized}}{\#\text{words}}
\]

\[
\text{WRR-1C} = \frac{\#\text{words with at most one wrong character}}{\#\text{words}}
\]

\[
\text{TRR} = \frac{\#\text{text images correctly recognized}}{\#\text{text images}}
\]

Evaluation of the proposed methods

Depending on the models and features, we have proposed 3 methods:

1) DBN-AE-BLSTM: DBN auto-encoder + BLSTM.
2) MLP-AE-BLSTM: MLP auto-encoder + BLSTM.
3) ConvNet-BLSTM: ConvNet Classifier + BLSTM.

The ConvNet and the deep auto-encoders are trained on 88% of ArabCharSet and tested on the remaining 12%. The trained ConvNet reaches a character classification error rate of 3.8%. The DBN and the MLP-based auto-encoders, achieve respectively 2.5% and 1.2% of average squared errors of reconstruction. Each of ArabTrainText, ArabValidText and ArabTestText has hence three different feature representations. Each representation of ArabTrainText has been used to train a BLSTM-CTC network. The number of LSTM cells per layer has been set to 300, the learning rate has been set to $10^{-4}$, and we have used a momentum of 0.9.

The obtained results are reported in Table I. These results show high rates with the 3 features. The proposed deep neural networks are hence able to extract relevant features that are well used by the BLSTM-CTC. They also show that the ConvNet-BLSTM method outperforms the two other ones. This can be explained by the specificities of diacritics in the Arabic text. As shown in Figure 1.b, some characters are distinguished only by diacritic marks. Using the reconstruction paradigm, these marks are often reproduced with blurring effect which introduces confusions (especially between letters with 2 and 3 points). The ConvNet is a powerful classification model. It focuses only on predicting the correct character class and derives features that discriminate between characters, whereas auto-encoders extract features that allow reconstructing the whole input image, the character part but also the noise and the background. Obtained results show also that the DBN-AE-BLSTM outperforms MLP-AE-BLSTM. This is due to the fact that RBMs are pre-trained to explore high-level character structures which are then projected to the features space.

In addition, more than 71% of words are correctly recognized with ConvNet-BLSTM and this rate increases by 16 points if we consider the WRR-1C measure, i.e. 16% of the recognized words differs by only one character from the ground-truth. This suggests that a post-processing step based on a language model can significantly improve the recognition rate at the word level.

<table>
<thead>
<tr>
<th>Method</th>
<th>CRR (%)</th>
<th>WRR (%)</th>
<th>WRR-1C (%)</th>
<th>TRR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN-AE-BLSTM</td>
<td>90.73</td>
<td>59.45</td>
<td>81.91</td>
<td>39.39</td>
</tr>
<tr>
<td>MLP-AE-BLSTM</td>
<td>88.50</td>
<td>59.95</td>
<td>79.02</td>
<td>33.19</td>
</tr>
<tr>
<td>ConvNet-BLSTM</td>
<td>94.36</td>
<td>71.26</td>
<td>86.77</td>
<td>55.03</td>
</tr>
</tbody>
</table>

Impact of the number of LSTM cells

We have also studied the influence of the number of LSTM cells per hidden layer on the recognition performance. For this, several BLSTM architectures have been considered in the method DBN-AE-BLSTM. The obtained results, shown in Figure 7, indicate that a maximum CRR is reached with

---

Footnote: Text images used to train and test the BLSTM schema are part of our ALIF dataset. This dataset will be publicly available soon.
300 or 350 cells, with no significant improvement with more cells. Thus, all of our experiments have been carried out using BLSTMs with 300 hidden cells in both directions. The number of 300 achieves the best trade-off between the recognition performance and computation complexity.

Fig. 7: Impact of the number of LSTM cells.

Learned features vs. hand-crafted ones
In order to evaluate the advantage of learned features, we have compared our methods to hand-crafted features-based approach as the one proposed in [12]. Hand-crafted features are computed as follows: a binarization step is applied on text images. A GMM classifier is trained to separate text and background classes. As in [12], 9 geometrical features are extracted per column from the resulting binary text images. A BLSTM-CTC has been trained using the obtained feature sequences. The resulting method is called HC-BLSTM. The comparative results are illustrated in Table II. The HC-BLSTM method is outperformed by the ConvNet-BLSTM one. As previously explained in Section III-A, learned features are built under the fixed goal of a better character representation. They give more valuable information to the BLSTM-CTC and cope with background complexity more than hand-crafted features.

Comparison with a commercial OCR
We have also performed a comparative study of the performance of our method w.r.t. an off-the-shelf OCR solution. We have chosen a well-known OCR engine, ‘ABBYY Fine Reader 12’[^4]. The Arabic OCR component of this engine has been applied on ArabTestText. Results have been evaluated using CRR and WRR. Our ConvNet-BLSTM method (and also the 2 other proposed methods) still outperforms the commercial solution with almost 11 points in terms of CRR (cf. Table II).

<table>
<thead>
<tr>
<th>Method</th>
<th>CRR (%)</th>
<th>WRR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvNet-BLSTM</td>
<td>94.36</td>
<td>71.26</td>
</tr>
<tr>
<td>HC-BLSTM</td>
<td>85.44</td>
<td>52.13</td>
</tr>
<tr>
<td>ABBYY</td>
<td>83.26</td>
<td>49.80</td>
</tr>
</tbody>
</table>

[^4]: http://finereader.abbYY.com/professional/

Table II: Comparative study.

V. CONCLUSION
We have presented in this study three methods for Arabic text recognition in videos. Each approach uses a deep learned model to represent the text image as a sequence of learned features. A BLSTM-CTC network sequentially analyze these features in order to predict the text transcription without any prior data segmentation. The obtained results highlight the good recognition rates of our methods that outperform an existing well-known commercial OCR systems. In addition, the chosen learned features are well-suited for this task and outperform hand-crafted ones. In a future work, we will address the use of a language model to further enhance the recognition rates.

REFERENCES