Multi-dimensional long short-term memory networks for artificial Arabic text recognition in news video

Oussama Zayene\textsuperscript{1,2}  \textsuperscript{a}, Sameh Masmoudi Touj\textsuperscript{1}, Jean Hennebert\textsuperscript{3}, Rolf Ingold\textsuperscript{2}, Najoua Essoukri Ben Amara\textsuperscript{1}

\textsuperscript{1}LATIS Laboratory, National Engineering School of Sousse, University of Sousse, Tunisia
\textsuperscript{2}DIVA Group, University of Fribourg, Switzerland
\textsuperscript{3}Institute of Complex Systems, HES-SO, University of Applied Science Western Switzerland, Switzerland
\textsuperscript{a}E-mail: oussama.zayene@unifr.ch

Abstract: This study presents a novel approach for Arabic video text recognition based on recurrent neural networks. In fact, embedded texts in videos represent a rich source of information for indexing and automatically annotating multimedia documents. However, video text recognition is a non-trivial task due to many challenges like the variability of text patterns and the complexity of backgrounds. In the case of Arabic, the presence of diacritic marks, the cursive nature of the script and the non-uniform intra/inter word distances, may introduce many additional challenges. The proposed system presents a segmentation-free method that relies specifically on a multi-dimensional long short-term memory coupled with a connectionist temporal classification layer. It is shown that using an efficient pre-processing step and a compact representation of Arabic character models brings robust performance and yields a low-error rate than other recently published methods. The authors’ system is trained and evaluated using the public ActIV-R dataset under different evaluation protocols. The obtained results are very interesting. They also outperform current state-of-the-art approaches on the public dataset ALIF in terms of recognition rates at both character and line levels.

1 Introduction

Since the 1980s, research in optical character recognition (OCR) techniques has been an attractive area in computer vision and pattern recognition communities [1, 2]. Prior studies have addressed specific research problems that bordered on handwritten texts [3, 4] and scanned documents in domains such as postal address [5] and bank cheques [6] recognition.

For the two last decades, embedded texts in videos and natural scene images have received increasing attention as they often give important information about the multimedia content [7–9]. Recognising text in videos, often called video OCR, is an essential task in a lot of applications like video indexing and content-based multimedia retrieval. Most existing video OCR systems are dedicated to few languages, such as Latin and Chinese [7–11]. For a language like Arabic, which is used by more than 500 million people around the world, such systems are much less developed.

A video OCR system is basically composed of two main phases: text localisation, which may include detection and tracking of text regions; and text recognition, which may include extraction, segmentation, and recognition of already detected text regions. In this study, we focus on the second phase by proposing a new approach for recognising Arabic text in the news video.

Compared with the case of printed/handwritten text recognition in scanned documents, video text recognition is more challenging due to many factors including

- Background complexity: the presence of text-like objects such as fences and bricks can be confused with text characters.
- Text patterns variability: text in videos mostly has an unknown size/font and differs in colour and alignment.
- Video quality: it includes acquisition conditions, compression artefacts, and low resolution.

Fig. 1a provides examples of frames collected from different news TV channels for typical problems in video text recognition. Text in videos is generally classified into scene text and artificial (or superimposed) text. The first type is naturally recorded as part of a scene during video capturing, like text on signs and clothing, and may include handwritten material. The second type is artificially superimposed on the video during the editing process. Fig. 1b illustrates a video frame from TunisiaNat1 TV including scene text in the form of a traffic sign, and artificial text in the form of subtitles describing event information. Compared with the scene text, the artificial one can provide a brief and direct description of video content, which is important for automatic broadcast annotation. Typically artificial text in a news video indicates the names of people, event information, location, scores of a match, etc. In this context, we particularly focus on this type of text.

The recognition of Arabic texts for indexing Arabic documents has recently become a compelling research domain. Several techniques have been proposed in the conventional field of Arabic OCR in scanned documents [4, 12–15]. However, very few attempts have been made on the development of recognition systems for artificial text in Arabic news videos [16, 17], despite the presence of several Arabic news channels with very high viewing rates in the Arabic world and outside of it. Actually, we need to extract embedded texts from the video content as powerful semantic cues for automatic broadcast annotation. Compared with Latin text, the Arabic one has special characteristics.

- It is cursive with high connectivity between characters, i.e. most of them have right and/or left connection points linked to the baseline.
- Arabic characters can have up to four shapes depending on their position in the word: at the beginning, in the middle, at the end or isolated. Fig. 1c presents a decomposition example of an Arabic word into individual characters, labelled in Latin and accompanied by suffixes indicating their positions.
- The spaces between pieces of Arabic words are not uniform and vary in size, making ambiguities to distinguish between stroke ends or word ends in the segmentation phase.
- Arabic characters may have exactly the same shape and are distinguished from each other only by a diacritic mark, which may appear above or below the main character such as letters
The rest of the manuscript is organised as follows. The related work is reviewed in Section 2. Section 3 presents a short overview of RNN networks with a main focus on the LSTM architecture. The proposed system is presented in Section 4. Section 5 describes the grouping strategy of character models. The used benchmark datasets and the obtained experimental results are provided in Section 6. Section 7 draws the conclusions and outlines the future work.

2 Related work

Video text is usually embedded in complex backgrounds with different colours, scales, and fonts, which makes it difficult to be recognised by means of a standard OCR engine. According to the literature, there are essentially two ways to solve this problem, which are: (i) recognising characters by separating text pixels from the background beforehand and then applying an available OCR software [24–27]. (ii) Recognising characters by using features and classifiers specially designed for video or/and natural scene text [28–31].

The first methodology requires an appropriate pre-processing stage to obtain characters with well-defined shapes and a plain background. Several techniques proposed a robust image binarisation for this aim. For instance, Zhou et al. [24] suggested an edge-based method for binarising text in video images. Zhang and Wang proposed a method [25] for binarising artificial text in the video using K-means algorithm in the RGB space with a Markov random field model, and Log-Gabor filters as a refinement step. Similarly, Hu et al. [26] put forward a binarisation method for both overlaid and scene texts utilising two confidence maps and K-means clustering algorithm. Roy et al. [27] introduced a new method to binarise video text based on a Bayesian classifier for text/non-text pixels discrimination and a connected component analysis for text information restoring. After obtaining the binarised text image, these methods made use of the Google's OCR engine Tesseract for recognition. However, in this kind of methodology, the recognition performance usually relies on the efficiency of the text binarisation and may suffer from noise and distortion in complex backgrounds.

On the other hand, the second methodology uses classifiers directly on text regions mixed with background objects. For example, Zhai et al. [28] put forward a segmentation-free method based on bidirectional RNNs (BRNNs) with a CTC layer for Chinese news text recognition. To train this network, the authors collected 2 million news titles from 13 TV channels. Su and Lu [29] extracted sequences of histogram of oriented gradient (HOG) features as a sequential image representation and generated the corresponding character sequence with RNNS. Jünger et al. [30] proposed a convolutional neural network (CNN)-based classifier to holistically recognise words in natural images. The utilised deep neural models were trained on a large scale synthetic dataset. Recently, some published work [7, 16, 31, 32] have jointly used the CNN and RNN for recognising text in natural scene images or/and videos. These methods are generally composed of two modules, a deep CNN for feature extraction and a BRNN for sequence modelling. In [31], video texts were first represented as sequences of learned features with a multi-scale scanning scheme. The sequence of features was then fed into a connectionist recurrent model, which would recognise text words without prior binarisation or explicit character segmentation. Shi et al. [32] treated word recognition as a sequence labelling problem. CNNs and BRNNs were employed to, respectively, extract feature sequences from the input images and generate sequential labels of arbitrary length. The CTC was adopted to decode the sequence. Wang et al. [7] explored a GMM-HMM bootstrap model to align the frame sequence with the transcription. The alignments were then utilised as supervision to train the CNN. BRNNs were finally used to model the text sequences. In fact, this kind of methodology usually requires a large number of samples covering various text fonts and backgrounds to train the classifier. Like the document-based OCR techniques, video OCR can also be divided into segmentation-based and segmentation-free recognition methods. The former, known as an analytical approach,
segments the lines, words or sub-words into smaller units (characters or graphemes) for recognition. The latter, also called global approach, takes the whole line or word image for recognition. In the case of Arabic text, the recognition systems have traditionally been segmentation-based. For instance, Ben Halima et al. [17] proposed to recognise Arabic video texts using an analytical approach. Text lines were first binarised and then segmented by projection profiles. Characters were finally classified using the fuzzy k-nearest neighbours (KNN) algorithm applied on a set of handcrafted features including occlusions, diacritic positions, projection profiles and a number of foreground-to-background transitions. In the same vein, Iwata et al. [33] adopted such methodologies to recognise Arabic texts in news video frames. Text lines were first segmented into words utilising a space detection algorithm. The character candidates were then over-segmented into primitive segments. The recognition was finally performed using the modified quadratic function classifier at the character level and the dynamic programming at the word level. With such approaches, the segmentation errors can propagate further and impact the recognition performance.

Yet, segmentation-free methods recognise a succession of characters directly from the text image, without any explicit segmentation. Such systems are based on classifiers like HMMs [18], [34] or RNNs [35]. In [16], [36], three RNN-based systems were proposed for Arabic video text recognition. These systems differed by their feature extraction scheme and had a common classifier. Firstly, a multi-scale sliding window was used to extract features based on three different feature-learning models. The two first models made use of deep auto-encoders, whereas the third one was CNN-based. A bidirectional LSTM (BLSTM) network coupled with a CTC output layer is afterwards used to predict correct characters of the input text image from the associated sequence of features without pre-segmented data. Naz et al. [37] suggested an RNN-based system for Urdu Nasta’liq (Urdu is a derivative of the Arabic alphabet) text recognition. The input textline images were first normalised to a fixed height, then transformed into a sequence of manually-designed features including horizontal and vertical projections, pixel distribution features and grey-level co-occurrence matrix (GLCM) features (contract, energy, correlation, and homogeneity). These features were next fed to the RNN in a frame-wise fashion, followed by a CTC decoding layer that transcribed the input data and finally provided the recognised sequence.

Most of the aforementioned Arabic text recognition systems rely on a feature extraction process. Nevertheless, feature design is a challenging and time-consuming task due to its dependence on the domain knowledge and past experience of human experts [14], [38]. On the other hand, there has been recent work that proposes recognition systems performing automatic feature extraction inside a learning scheme that operates directly on the raw pixel data. Such systems have shown high performance on different OCR tasks [15], [22], [23] and received considerable attention, especially with the resurgence of LSTM-RNNs. A comparison between the results of two recent work [38], [39] for Arabic handwriting recognition shows that a one-dimensional (1D) LSTM network operating on raw image pixels [39] outperforms the same network trained using whether handcrafted or learned features [38]. Motivated by this observation, we propose to recognise Arabic video text without relying on an explicit feature extraction stage. This is done by applying a multi-dimensional LSTM-RNN architecture on the input sequence.

3 Overview of RNNs

RNNs were first introduced in the 1980s and have become popular due to their ability to model contextual information. They represent powerful tools for processing patterns occurring in time series. In its simplest form, an RNN is an multi-layer perceptron (MLP) with recurrent layers.

Consider an input sequence $x$ presented to an RNN with $I$ input units, $H$ hidden units, and $K$ output units. Then the hidden units $a_h$ and the activations $b_o$ of a recurrent layer are calculated using the following equations:

$$a_h(t) = \sum_{i=1}^{I} w_{ih} x_i(t) + \sum_{h=1}^{H} w_{hh} a_h(t-1),$$

$$b_o(t) = \Theta_3 a_h(t),$$

where $x_i$ is the value of an input $i$ at time $t$, $a_h(t)$ and $b_o(t)$, respectively, denote the network input to a unit $i$ and the activation of unit $j$ at time $t$. $w_{ij}$ denotes the connection from a unit $i$ to a unit $j$, and $\Theta_3$ is the activation function of a hidden unit $h$.

Robinson [20] was among the first who suggested the use of standard RNNs for speech recognition. Lee & Kim [40] and Senior & Robinson [19] applied such networks to handwriting recognition.

In 1997, Schuter and Paliwal [18] introduced BRNNs by implementing two recurrent layers, one processing the sequence in a forward direction (left to right) and the other backwards. Both layers are connected to the same input and output layers.

The multi-dimensional recurrent neural network (MDRNN) architecture [41] represents a generalisation of RNNs, which can deal with multi-dimensional data, e.g. image (2D), video (3D) etc. To extend the RNN to a multi-dimensional RNN, let $p \in \mathbb{Z}^d$ be a point in an $n$-dimensional input sequence $x$ of dimensions $D_0, \ldots, D_n$. Instead of $a(t)$ in a 1D case, we write $a^p$ as an input in the multi-dimensional case. The upper index $p$, $i \in \{1, 2, 3, \ldots, n\}$, is used to define the position. $D_i = (p_1, \ldots, p_d - 1, \ldots, p_n)$ denotes the position on a step back in dimension $d$. Let $w_{ij}^p$ be the recurrent connection from $i$ to $j$ along dimension $d$. The forward equation for an $n$-dimensional MDRNN is calculated according to the following equations:

$$d_h^p = \sum_{i=1}^{I} w_{ih} x_i^p + \sum_{d=1}^{n} \sum_{h=1}^{H} w_{jd}^p w_{dh},$$

$$b_o^p = \Theta_3 (d_h^p).$$

The backward pass is given by (3), where $e^p = \frac{\partial E}{\partial b_o^p}$ and $\delta^p = \frac{\partial E}{\partial a_h^p}$, respectively, denote the output error of the unit $j$ at time $p$ and the error after accumulation

$$e_h^p = \sum_{k=1}^{K} \delta_k^p w_{hk} + \sum_{d=1}^{n} \sum_{i=1}^{I} \frac{\partial E}{\partial a_i} w_{ih}^p,$$

$$\delta_h^p = \Theta_3'(a_h^p) e_h^p.$$

While standard RNNs use a recurrence only over 1D, like the x-axis of an image, the MDRNNs can scan the input image along both axes, allowing the exploitation of more context and the modelling of the text variations in four directions (left, right, top, and bottom). In particular, the 2DRNN forward pass starts at the origin $(0, 0)$, follows the direction of the arrows and scans through the 2D input sequence $X^0$, as illustrated in Fig. 2. It is to be noted that the point $(i,j)$ is never reached before both $(i-1, j)$ and $(i, j-1)$ [42].

3.1 LSTM networks

The problems of long-term dependencies and vanishing gradient – the gradient of the loss function decays exponentially over time [43] – were the reason for the lack of practical applications of RNNs. In 1997 [44], an advance in designing such networks was introduced as the long short-term memory (LSTM). LSTM networks are a special class of RNNs that use memory cells as hidden layer units. These cells can maintain information for long periods of time.

LSTM consists of a set of three multiplicative gates, so-called the input gate $i$, the output gate $o$, and the forget gate $f$, to control when information should be stored or removed from the memory cell $c$. This architecture lets them learn longer-term dependencies (see Fig. 3b for an illustration). LSTM first computes its gates' activation $i_t$ (4), $f_t$ (5), and updates its cell state from $c_{t-1}$ to $c_t$ (6).
4 Proposed system

The proposed video text recognition system is based on an MDLSTM network coupled with a CTC output layer. It is mainly developed using an adapted version of the open-source RNNLib toolkit. The use of RNNLib goes typically through two steps: training and test. During the training step, the network learns the sequence-to-sequence matching in a supervised fashion, i.e. the alignment between the input and the output sequences. In the test step, the normalised textline image is fed to the trained MDLSTM model, which generates the predicted sequence. For both steps, we apply the same pre-processing operations.

In what follows, we describe the pre-processing stage.

4.1 Pre-processing

As mentioned before, the video OCR domain has many problems to deal with in regard to the variability of text patterns, the complexity of backgrounds, etc. Therefore, we propose to apply some pre-processing prior to the recognition step in order to reduce these undesirable effects. Given a textline image, pre-processing steps of text polarity normalisation and image size scaling are performed. First, the text polarity is determined; i.e. judging whether it is dark text on light background or vice versa, using a skeleton-based technique. Skeletons are important shape descriptors in object representation and recognition. The generalised skeleton representation of a binary image is the union of sets \( S_i \) given by the following equation:

\[
S_i(X) = (X \ominus nB) - (X \ominus nB) \ast B,
\]  

where \( S_i(X) \) represent the skeleton subsets of a binary image containing a set of topologically open shapes \( X, n \) is the number of shapes, and \( B \) is a structuring element. The symbols \( \ominus \) and \( \ast \) refer to the binary erosion and opening, respectively. Note that the binary images \( Bin \) and \( Bin \) are obtained by adaptive thresholding the input greyscale image \( Gs \) and its negative version \( Gs \) (step (2) of Algorithm 1 (see Fig. 4)). It can be observed from the content distribution of the skeleton maps (steps (3) and (4) of Algorithm 1 (Fig. 4)) created with the correct gradient direction, that the skeleton pixels are retained in the centre line of the character shape (e.g. skeleton dark-on-light (DL) in Fig. 5a and skeleton light-on-dark (LD) in Fig. 5b). This is due to the characteristics of the skeleton function that generates a thin version of the original shape, which is equidistant to its boundaries. Otherwise, the skeleton pixels all surround the characters and are placed on the image boundaries (cf. skeleton LD in Fig. 5a and skeleton DL in Fig. 5b). Thus, the text gradient direction is simply obtained by comparing the number of white pixels (WPs) located on the boundaries of the two skeleton images (step (11) of Algorithm 1 (Fig. 4)), i.e. we invert the input greyscale image if its skeleton LD has fewer WPs on the boundaries (step (12) of Algorithm 1 (Fig. 4)). Subsequently, the text polarity is normalised to DL for all input greyscale images, as shown at the bottom of Fig. 5. This method has been able to achieve an accuracy of 95% on our dataset.

All the normalised images are then scaled to a common height (determined empirically) using the bi-linear interpolation method.

4.2 Network architecture

As depicted in Fig. 6, our network consists of five layers of which three are LSTM-based hidden layers (for each direction) and two are feedforward subsampling layers with \( tanh \) as an activation function. We adopt the hierarchical network topology as used in [42] by repeatedly composing MDLSTM layers with feedforward \( tanh \) layers. The purpose of the subsampling step is to compress the
sequence into windows, thus speeding up the training time with the MDLSTM architecture. The subsampling is also required for reducing the number of weight connections between hidden layers.

In this network, there are mainly four important parameters that require tuning during the training phase.

- The **input block size** refers to the ‘width × height’ of the pixel block used to initially divide the input text image into small patches. We empirically set the size of this parameter as 2 × 4 or 1 × 4 depending upon the evaluation protocol (see Section 5).
- The **LSTM size** refers to the number of LSTM cells in each hidden layer. In our work, 2, 10 and 50 represent the appropriate values for this parameter. These values are found empirically and they match as well those reported by other researchers [22, 37, 42]. Note that the number of LSTM cells, for each hidden layer, should be equal to the size of that layer multiplied by the number of directions in which the input image is scanned. In the proposed architecture, the image is scanned in four different directions. Hence, the number of LSTM cells becomes 2 × 4, 10 × 4 and 50 × 4. This is shown in Fig. 6 by four different colours of LSTM cells.
- The **tanh size** describes the number of tanh units in each subsampling layer.
- The **subsampling window size** refers to the window required for subsampling the input from each layer before feeding it to the next hidden layer. This parameter decreases the sequence length, in the applied layer, by a factor corresponding to the window width. The optimal sizes are set to 1 × 4 for both first and second hidden layers. At the hidden-to-output layer transition, no subsampling is applied.

### 4.3 CTC layer

The output of the last LSTM hidden layer is passed to a CTC output layer, which is used as an output layer with softmax activation function. This layer permits working on an unsegmented input sequence, which is not the case for standard RNN objective functions. The principle of such a layer is inspired by the forward-backward algorithm of the HMM [45] and is used to align the target labels with the LSTM output sequences. During training, this alignment enables the network to learn the relative location of labels in the whole transcription. The CTC layer contains as many units as there are elements in the alphabet L of labels, plus one extra ‘blank’ unit, i.e. the output alphabet is \( L' = L \cup \{ \text{\textasciitilde} \} \). ‘Blank’ is not a real character class, but a virtual symbol used to separate the consecutive real character. Let \( \pi \) be an input sequence of length \( T \) and \( \beta: L' \rightarrow L^2 \) be a mapping function, which removes duplicates then blanks in the network prediction. For example: \( \beta(a \text{\textasciitilde} ab) = \beta(aa \text{\textasciitilde} abb) = aab \). Since the network outputs for different time steps are conditionally independent given \( x \), the probability of a label sequence \( \pi \in L^T \) in terms of LSTM outputs is calculated according to the following equation:

\[
p(\pi|lx) = \prod_{t=1}^{T} y_{lt}(x),
\]

where \( y_{lt} \) is the activation of output unit \( k \) at time \( t \). The mapping \( \beta \) allows calculating the posterior probability of a character sequence \( \pi \in L^T \), which is the sum of the probabilities of all paths (\( L^T \)) corresponding to it

\[
p(l|lx) = \sum_{\pi \in \beta^{-1}(l)} p(\pi|lx).
\]

This ‘collapsing together’ of different paths to the same labelling is what allows the CTC to use unsegmented data. After that, the CTC objective function maximises the probability to find the most probable label sequence for the corresponding unsegmented training data \( S = \{(x, z), z \in L^n\} \) by minimising the following cost:

\[
\theta = -\sum_{(l, z) \in S} \log p(z|lx).
\]
same basic character Taa, so we obtain two samples of the model Taaa B instead of having one for the model Taaa B and another for the model Taaa M.

- Set72: we used here one single model for each character of Fig. 7, regardless of its position in the word.

The question to address regarding these sets is ‘does a trade-off exist between having more models per character (to capture the intrinsic details of each glyph, i.e. set165) and having more training samples per character model (without considering the details of character shapes, i.e. set104 and set75)?’
that refers to the letter’s position in the word, i.e. B: Begin, M: Middle, E: End, and I: Isolate. A typical example has been already provided at the line level for each text image. During the annotation process, 165 character shapes are considered, as are given in Table 1. To have an easily accessible representation of these metrics are defined as follows:

\[
\text{CRR} = \frac{\#\text{characters} - I - S - D}{\#\text{characters}},
\]

\[
\text{WRR} = \frac{\#\text{words\hspace{1pt}correctly\hspace{1pt}recognised}}{\#\text{words}},
\]

\[
\text{LRR} = \frac{\#\text{lines\hspace{1pt}correctly\hspace{1pt}recognised}}{\#\text{lines}}.
\]

### 6 Experiments

This section describes the set of experiments that we separately conducted to (i) fix the optimal network parameters, (ii) analyse the effect of both pre-processing and model sets on the recognition performance, and (iii) compare the proposed system with other recently published methods using two public datasets.

We now introduce the experimental setup in terms of data, evaluation protocols, and metrics.

#### 6.1 AcTiV-R dataset

To evaluate the proposed method, we use a part of our publicly available AcTiV database [47], namely AcTiV-R. It consists of 10,415 textline images, 44,583 words, and 259,192 characters distributed over four sets (one set per TV channel). Every set includes three sub-sets: ‘training-set’, ‘test-set’, and ‘closed-test set’ (used for competitions only [48]). As illustrated in Fig. 9, AcTiV-R texts are in various fonts and sizes, and with different degrees of background complexity. The recognition ground-truth is provided at the line level for each text image. During the annotation process, 165 character shapes are considered, as detailed above in the description of set165. We evaluate our work, specifically, in three protocols proposed by Zayene et al. [47]. More details about the protocols and statistics of the used dataset are given in Table 1. To have an easily accessible representation of Arabic text, it is transformed into a set of Latin labels with a suffix that refers to the letter’s position in the word, i.e. B: Begin, M: Middle, E: End, and I: Isolate. A typical example has been already depicted in Fig. 1c.

#### 6.2 Evaluation metric

The performance measure used in our experiments is based on the line recognition rate (LRR) and word recognition rate (WRR) at the line and word levels, respectively, and on the computation of insertion (I), deletion (D), and substitution (S) errors at the character level (CRR). These metrics are defined as follows:

\[\text{CRR} = \frac{\#\text{characters} - I - S - D}{\#\text{characters}},\]

\[\text{WRR} = \frac{\#\text{words\hspace{1pt}correctly\hspace{1pt}recognised}}{\#\text{words}},\]

\[\text{LRR} = \frac{\#\text{lines\hspace{1pt}correctly\hspace{1pt}recognised}}{\#\text{lines}}.\]
Note that for these experiments, we use the same network.

6.4 Impact of the pre-processing step:

6.4.1 Results and discussion

The bold values indicate the best values.

The experimental results of protocol (P6.3). An increase of 5.13% is achieved on the AllSD protocol (P6.2), from 62.44 to 67.73% for TunisiaNat1'sFrance24's protocol (P6.1), from 40.82 to 43.6% for RussiaToday's protocol (P6.4) and of 7% on the channel-free protocol (P9). The best results are marked in bold in Table 4.

6.4.2 Effect of model set choice: Table 5 provides the final obtained results on the AcTiV-R dataset: impact of model sets choice.

To examine the impact of the pre-processing step, we carry out several experiments by training two different types of input images, with and without text polarity normalisation. Note that for these experiments, we use the same network architecture and we fix the height of all images to 70 pixels. By carefully examining the obtained results given in Table 4, it is concluded that the pre-processing step has a clear beneficial effect on the recognition accuracy. The results indicate that by using both height and polarity normalisation, the LRR increases from 51.54 to 53% for AljazeeraHD's protocol (P3), from 51.40 to 57% for France24's protocol (P6.1), from 40.82 to 43.6% for RussiaToday's protocol (P6.2), and from 62.44 to 67.73% for TunisiaNat1's protocol (P6.3). An increase of 5.13% is achieved on the AllSD protocol (P6.4) and of 7% on the channel-free protocol (P9). The best results are marked in bold in Table 4.

6.4.4 Effect of model set choice: Table 5 provides the recognition results of set165, set104, and set72-based systems. We can see that the performances are increasing significantly (e.g. by 11.29% for P3) from set165 to set104. It seems beneficial to finely model the difference between begin-middle shapes and end-isolate ones. Intuitively, we should lose more precision of the modelling using fewer models. Nevertheless, we are probably observing here the effect of having too few training data for less frequent representations of some character shapes. For instance, the character TildAboveAlif (̀) in the position 'End' is represented with only 32 occurrences in the dataset. On the other hand, the performances decline considerably (at least 6%) from set104 to set72, where a single sub-model per character is used.

Overall, our best system for all evaluation protocols is the one based on the set104. The best accuracies are achieved on the TunisiaNat1 channel subset (P6.3) with 96.48% as a CRR and 72.49% as an LRR. An important increase of 9.4% for the channel-free protocol (P9) is achieved in terms of LRR.

6.4.5 Error analysis: Fig. 10 depicts some typical misrecognised images. It contains four blocks. Each one presents two (or three) input images and their corresponding output sequences. Block (a) shows two images from protocol 3. For each, we present its results with set165 and set104, respectively. As it can be seen, most erroneous characters in the first set are correctly recognised (green colour) using set104. Block (b) depicts two output lines (per image) of two different evaluation protocols, P6.1 and P6.4 (AllSD protocol). It is clear that for both images the results of P6.4 are better than those of P6.1. This can be explained by the presence of more training shapes in the AllSD protocol. Blocks (c) and (d)
present examples of output lines from P6.2 and P6.3, respectively. A visual inspection of the errors is actually supporting this statement, where frequent errors are related to less frequent shapes in the training database. Based on our knowledge about the specificities of Arabic alphabet, we divide the causes of errors into two categories: character similarity (substitution errors of block (a)) and insufficient samples of punctuation, digits and symbols (substitution and deletion errors of blocks (b), (c) and (d)). Several measures can be taken to minimise the character error rate. For instance, some errors can be corrected by integrating language models and dropout [22] to improve the LSTM-based recognition system and increase the generalisation performance [49].

6.4.4 Comparison with other methods: We validate here the performance of our proposed system by comparing it with the method presented by Iwata et al. [33] (see related work section).

As depicted in Fig. 11, we outperform Iwata's system by a large margin in all protocols. The obtained results, in terms of LRR, are higher with a gain ranging from 10 to 16% for protocols 6.2 and 6.3, respectively. It is to be noted that the current version of Iwata system is not compatible with high definition resolution.

We have also evaluated our system using a recently published dataset of superimposed video text recognition, namely ALIF [36]. The dataset is composed of 6532 cropped text images extracted from diverse Arabic TV channels, where 12% of them are from web sources. It contains two parts: one being the training dataset with 4152 text images and the other a test dataset with three subsets. ALIF [36] works only on standard definition (SD) resolution and presents 140 character shapes including digits and punctuation marks. Table 6 shows the comparative results for the proposed system against five recently proposed methods [16, 36]. Note that these systems have been developed by the same author that put forward the ALIF dataset [36], and four of them were BLSTM-based. For these experiments, we use the same pre-processing steps and optimal network parameters, which give us the best recognition accuracies on the AcTiV-R dataset. We also adopt the same rules of model grouping as those used for set104 in Section 5. Interestingly, our proposed MDLSTM network with the normalisation step outperforms the BLSTM systems whether they were based on manually crafted features (HC-BLSTM) or automatic learned features (DBN-AE-BLSTM, MLP-AE-BLSTM, and CNN-BLSTM). We are able to achieve results roughly 16% higher than the best rate obtained by the CNN-BLSTM system, in terms of LRR. These results are obtained on the ALIF Test1 subset [36], which includes 900 textline images.

7 Conclusion

We have presented in this study an Arabic video text recognition system based on an MDLSTM network coupled with a CTC output layer. The proposed system allows avoiding two hard OCR steps, which are a textline segmentation and feature extraction. The suggested method has been trained and evaluated using the AcTiV-
Table 6  Obtained results on ALIF dataset and comparison with others systems

<table>
<thead>
<tr>
<th>Method</th>
<th>CRR, %</th>
<th>WRR, %</th>
<th>LRR, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN-AE-BLSTM [16]</td>
<td>90.73</td>
<td>59.45</td>
<td>39.39</td>
</tr>
<tr>
<td>MLP-AE-BLSTM [16]</td>
<td>88.50</td>
<td>59.95</td>
<td>33.19</td>
</tr>
<tr>
<td>CNN-BLSTM [16]</td>
<td>94.36</td>
<td>71.26</td>
<td>55.03</td>
</tr>
<tr>
<td>HC-BLSTM [36]</td>
<td>85.44</td>
<td>52.13</td>
<td>—</td>
</tr>
<tr>
<td>ABBYY [36]</td>
<td>83.26</td>
<td>49.80</td>
<td>26.91</td>
</tr>
<tr>
<td>proposed system</td>
<td>96.85</td>
<td>85.71</td>
<td>70.67</td>
</tr>
</tbody>
</table>

The bold values indicate the best values.

8  Acknowledgments

The researchers would like to thank the developing team of RNNLib for providing such a fully functional and open-source library. The researchers would also like to thank all the TV channels for providing us with data and multimedia files, especially the archive staff of TunisiaNat1 TV.

9  References


© The Institution of Engineering and Technology 2018