IET Computer Vision

Research Article

IET Journals
The Institution of Engineering and Technology

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

ISSN 1751-9632
Received on 12th October 2017
Revised 12th February 2018
Accepted on 13th March 2018
doi: 10.1049/iet-cvi.2017.0468
www.ietdl.org

Oussama Zayene[1,2] ✉, Sameh Masmoudi Touj[1], Jean Hennebert[3], Rolf Ingold[2], Najoua Essoukri Ben Amara[1]

[1]*LATIS Laboratory, National Engineering School of Sousse, University of Sousse, Tunisia*
[2]*DIVA Group, University of Fribourg, Switzerland*
[3]*Institute of Complex Systems, HES-SO, University of Applied Science Western Switzerland, Switzerland*
✉ *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/handwriting 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. 1*a* 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. 1*b* 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. 1*c* 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

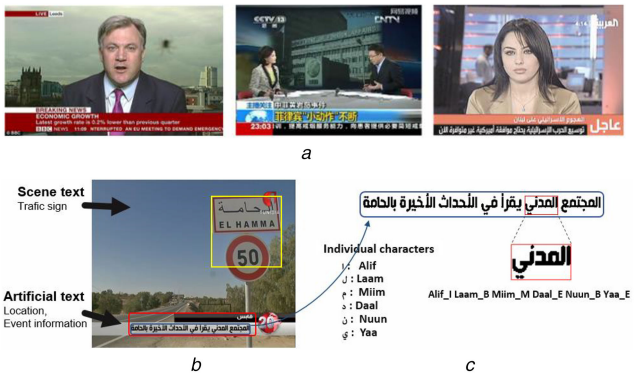

Fig. 1 *Frame samples from different TV channels depicting typical characteristics of video text*

(a) From left to right: examples of BBC, CCTV, and Al-Arabiya TV news channels, ***(b)*** Video frame displaying Arabic scene and artificial texts, ***(c)*** Decomposition example of an extracted word into characters

Baa (ب), *Taaa* (ت) and *Thaa* (ث). These diacritics are normally a dot, a group of dots (two or three), a *Hamza* (ء) or a *Tild* (~). It is worth noting that any deletion or erosion of these diacritic marks results in a misrepresentation of the character. Hence, any binarisation algorithm needs to efficiently deal with these dots so as not to change the identity of the character.

- Arabic has several standard ligatures, which are formed by combining two or more letters. The most common one is *LaamAlif* (لا), which is a combination of *Laam* (ل) and *Alif* (ا).

More details on the specificities of the Arabic script can be found in [4, 12]. All these characteristics along with the previously mentioned challenges may give rise to failures in Arabic video text recognition tasks.

In this study, we propose a novel text recognition system based on a segmentation-free method, which relies on the use of LSTM networks. These networks have been successfully applied in different sequence classification problems and have outperformed alternative hidden Markov models (HMMs) [13, 14], recurrent neural networks (RNNs) [18, 19], their combination [20], and other sequence learning models. Some benchmark work has been developed using the RNN-LSTM networks, such as handwriting recognition of Latin and Asian scripts [21–23]. The excellent performance of such networks has motivated us to investigate their application for the recognition of Arabic video text.

The major contributions of this study are the following:

- Up to our knowledge, we are the first to apply the multi-dimensional LSTM (MDLSTM) architecture [15] for Arabic video text recognition. Such an architecture is adopted to model the text variations on both axes of the input image.
- We propose a new and simple pre-processing step that consists of text polarity normalisation utilising a skeleton-based technique. This pre-processing contributes to improving the performance of our system, as it will be shown in the experimental section.
- We suggest a compact representation of character models by applying a regrouping process of character shapes, benefiting from the morphological characteristics of the Arabic language. This representation has a direct impact on the size of the connectionist temporal classification (CTC) output layer of the used network.

To sum up, we put forward an innovative scheme based on the combination of a new pre-processing, a compact representation of character models and an application of the MDLSTM network to recognise Arabic text lines without any prior segmentation or binarisation steps. We aim also in this work to stand out from the dominant methodology, based on the so-called handcrafted features. This is done by applying an MDLSTM network that automatically learns features from the raw input image.

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. Jaderberg *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 [13, 14, 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_h of a recurrent layer are calculated using the following equations:

$$\begin{aligned} a_h(t) &= \sum_{i=1}^{I} w_{ih} x_i(t) + \sum_{h'=1}^{H} w_{h'h} b_{h'}(t-1), \\ b_h(t) &= \Theta_h(a_h(t)), \end{aligned} \tag{1}$$

where $x_i(t)$ is the value of an input i at time t, $a_j(t)$ and $b_j(t)$, respectively, denote the network input to a unit j and the activation of unit j at time t. w_{ij} denotes the connection from a unit i to a unit j, and Θ_h 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 $D_1, \ldots, D_n$. 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, $i \in \{1, 2, 3, \ldots, n\}$, is used to define the position. $P_d^- = (p1, \ldots, p_d - 1, \ldots, p_n)$ denotes the position on a step back in dimension d. Let w_{ij}^d 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:

$$\begin{aligned} a_h^p &= \sum_{i=1}^{I} w_{ih} x_i^P + \sum_{d=1}^{n} \sum_{h'=1}^{H} b_h^{P_d^-} w_{h'h}^d, \\ b_h^p &= \Theta_h(a_h^p). \end{aligned} \tag{2}$$

The backward pass is given by (3), where $\varepsilon_j^p = \partial E / \partial b_j^p$ and $\delta_j^p = \partial E / \partial a_j^p$, respectively, denote the output error of the unit j at time p and the error after accumulation

$$\begin{aligned} \varepsilon_h^p &= \sum_{k=1}^{K} \delta_k^p w_{hk} + \sum_{d=1}^{n} \sum_{h'=1}^{H} \delta_h^{p_d^+} w_{h'h}^d, \\ \delta_h^p &= \theta'_h(a_h^p) \varepsilon_h^p. \end{aligned} \tag{3}$$

While standard RNNs use a recurrence only over 1D, like the x-axis of an image, the MDRNNs 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^p, 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. 3*b* 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).

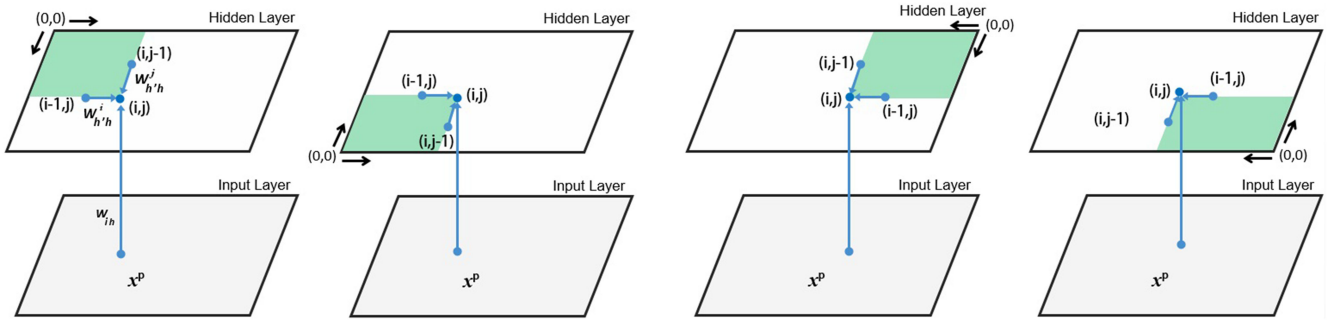

Fig. 2 *Scanning directions of 2D RNN, inspired from [42]*

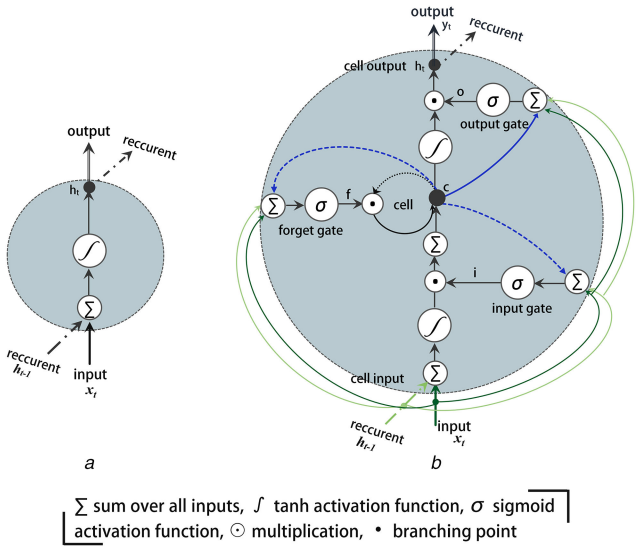

Fig. 3 *Detailed schematic of neurons for RNNs*
(a) Simple neuron, ***(b)*** LSTM unit

It then computes the output gate activation o_t (7) and finally outputs a hidden representation h_t (8). The inputs of an LSTM unit are the observations x_t and the hidden representation from the previous time step h_{t-1}. LSTM runs the following set of update operations:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + W_{ic}c_{t-1} + b_i), \tag{4}$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + W_{fc}c_{t-1} + b_c), \tag{5}$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \tag{6}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + W_{oc}c_t + b_o), \tag{7}$$

$$h_t = o_t \tanh(c_t), \tag{8}$$

where W denotes the weight matrices, b denotes the bias vectors and $\sigma(x) = 1/1 + e^{-x}$ is the logistic sigmoid function.

Standard LSTM is explicitly 1D since each cell contains a single recurrent connection, whose activation is controlled by a single forget gate. Nevertheless, it is possible to extend this to n dimensions, i.e. an MDLSTM memory cell, by using n recurrent connections instead (one for each of the cell's previous states along every dimension), with n forget gates.

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 $\{S_n\}$ given by the following equation:

$$S_n(X) = (X \ominus nB) - (X \ominus nB) \circ B, \tag{9}$$

where $S_n(X)$ 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 $\circ$ refer to the binary erosion and opening, respectively. Note that the binary images Bin and $\overline{Bin}$ are obtained by adaptive thresholding the input greyscale image Gs and its negative version $\overline{Gs}$ (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. 5*a* and skeleton light-on-dark (LD) in Fig. 5*b*). 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. 5*a* and skeleton DL in Fig. 5*b*). 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

Input : original text image *In*
Output: normalized image
Gs ← rgb-To-grayscale(In)
Bin← Binarization (Gs)
S_i ← SkeletonExtract (Bin) *// using equation (9)*
$\overline{S_i}$← SkeletonExtract ($\overline{Bin}$) *// using equation (9)*
for ***all*** *pixel* $\boldsymbol{I}(x, y)$ *in image* S_i **do**
  if $(\boldsymbol{I}(x, y) \in$ *border of* S_i *&* $is > 0)$ **then**
    increase WP by 1
  end
end
Repeat steps (5 - 9) for image $\bar{S}_i$ to compute $\mathrm{WP}_{\bar{S}}$
if $WP_{\bar{S}} > WP_S$ **then**
  Text polarity inversion to DL
else
  No inversion of text polarity
end

Fig. 4 *Algorithm 1: Text polarity normalisation to DL*

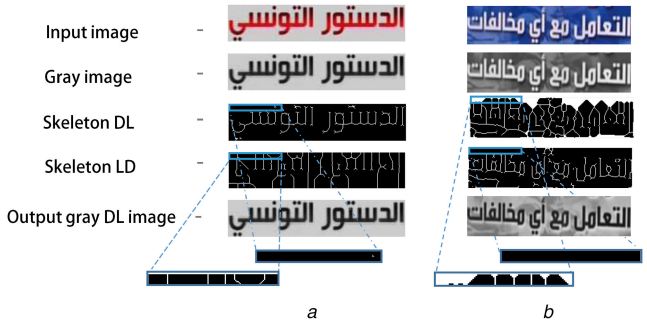

Fig. 5 *Pre-processing step of text gradient normalisation*
(a) DL text, ***(b)*** LD text

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 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 become 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 $L' = L \cup \{\varnothing\}$. 'Blank' is not a real character class, but a virtual symbol used to separate the consecutive real character. Let x be an input sequence of length T and $\beta: L' \to L^{\leq T}$ be a mapping function, which removes duplicates then blanks in the network prediction. For example: $\beta(a\varnothing\ \varnothing ab) = \beta(aa\varnothing\ \varnothing abb) = aab$. 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:

$$p(\pi|x) = \prod_{t=1}^{T} y^t_{\pi t}(x), \tag{10}$$

where y^t_k is the activation of output unit k at time t. The mapping β allows calculating the posterior probability of a character sequence $l \in L^{\mathrm{T}}$, which is the sum of the probabilities of all *paths* ($L^{'\mathrm{T}}$) corresponding to it

$$p(l|x) = \sum_{\pi \in \beta^{-1}(l)} p(\pi|x)\,. \tag{11}$$

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 $S = \{(x, z), z \in L^{\mathrm{lxl}}\}$ by minimising the following cost:

$$\vartheta = -\sum_{(x,z) \in S} \log p(z|x)\,. \tag{12}$$

5 Choice of model sets

By a model set, we mean the number of classes used to represent the different variations in character shapes. Benefiting from the morphological characteristics of the Arabic alphabet, we propose a glyph-based grouping method similar to [46], which leads to three sets with, respectively, 165, 104 and 72 classes, as described below. This proposal has a direct impact on the size of the CTC output layer, and consequently on the behaviour of the network.

- Set165: As stated in the Introduction, the Arabic alphabet contains 28 characters and most of them change shape according to their position in the word. Taking into account this variability, the number of shapes increases from 28 up to 100. In addition, the Arabic script includes two groups of extra characters. The first one represents a variation in some basic characters like the *TaaaClosed* (ۃ), which is a special form of the character *Taaa* (ت), and the *HamzaAboveWaaw* (ؤ), a combination of *Hamza* (ء) + *Waaw* (و). The second group includes four ligatures created when the character *Alif* (or one of its variants) follows the character *Laam* in the word. Considering these extra characters, there are overall 125 shapes (see Fig. 7 for examples). Adding to that, 10 digits, 13 punctuation marks and 12 additional characters that are combined with the diacritic mark *Chadda* (ّ), the total number of models goes up to 165.
- Set104: Using *set165*, we group similar glyphs into 104 models according to the following rules: (1) 'beginning' and 'middle' shapes share the same model. (2) 'End' and 'isolated' shapes share the same model. These rules are applied to all alphabet characters except for the characters *Ayn* (ع) and *Ghayn* (غ) where the initial, middle, final, and isolated shapes are too different. This strategy of grouping is natural as beginning-middle and end-isolated character shapes are visually similar. For instance, the two first character models (left-to-right) of the word in Fig. 8 are grouped to one model as they belong to the

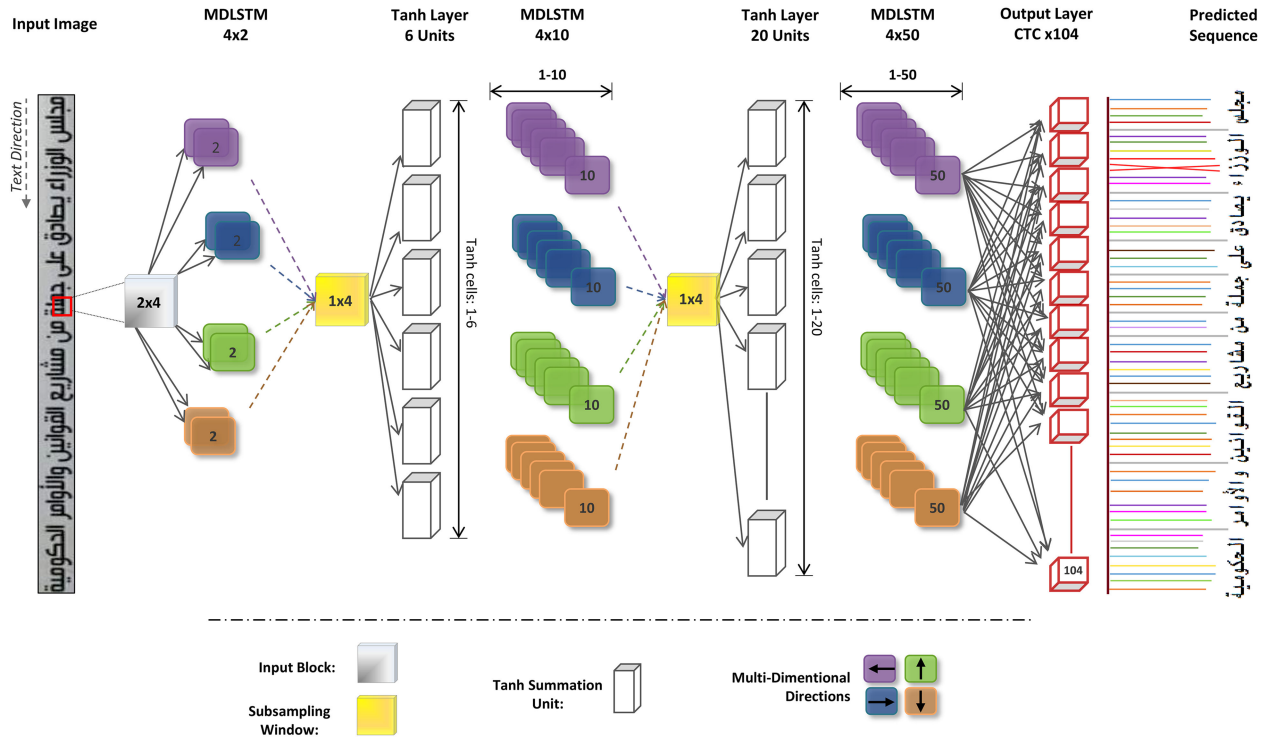

Fig. 6 *Architecture of the used subsampling hierarchical MDLSTM Network. Different colours indicating four scan directions of LSTM layers*

Character label	In Arabic I	B	M	E
❶ Arabic Alphabet				
Alif	ا	-	-	ا
Baa	ب	بـ	ـبـ	ـب
Thaa	ث	ثـ	ـثـ	ـث
Jiim	ج	جـ	ـجـ	ـج
Haaa	ح	حـ	ـحـ	ـح
Daal	د	-	-	ـد
Zaay	ز	-	-	ـز
Siin	س	سـ	ـسـ	ـس
Daad	ض	ضـ	ـضـ	ـض
Thaaa	ط	طـ	ـطـ	ـط
Ayn	ع	عـ	ـعـ	ـع
Ghayn	غ	غـ	ـغـ	ـغ
Kaaf	ك	كـ	ـكـ	ـك
Miim	م	مـ	ـمـ	ـم
Nuun	ن	نـ	ـنـ	ـن
Yaa	ي	يـ	ـيـ	ـي
❷ Extra Characters				
HamzaAboveAlif	أ	-	-	ـأ
TildAboveAlif	آ	-	-	ـآ
HamzaUnderAlif	إ	-	-	ـإ
HamzaAboveWaaw	ؤ	-	-	ـؤ
AlifBroken	ى	-	-	ـى
HamzaAboveAlifBroken	ىء			ـىء
Hamza		ء		
TaaaClosed	ة	-	-	ـة
❸ Ligatures				
LaamAlif	لا	-	-	ـلا
LaamHamzaAboveAlif	لأ	-	-	ـلأ
LaamHamzaUnderAlif	لإ	-	-	ـلإ
LaamTildAboveAlif	لآ	-	-	ـلآ
❹ Additional Characters				
WaawChadda	وّ	-	-	ـوّ
RaaChadda	رّ	-	-	ـرّ
LaamChadda	لّ	لّـ	ـلّـ	ـلّ
SiinChadda	سّ	سّـ	ـسّـ	ـسّ
❺ Digits				
Digit_0		0		
Digit_1		1		
Digit_2		2		
Digit_9		9		
❻ Punctuation marks				
Point		.		
Colon		:		
Comma		,		
Slash		/		
Percent		%		
Question		?		
Exlamation		!		
Hyphen		-		
Add		+		
Quote		"		
ParenthesisO		(		
ParenthesisC		)		
Bar		|		
SPACE				

Fig. 7 *Different class models to be recognised*

same basic character *Taaa*, 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)'.

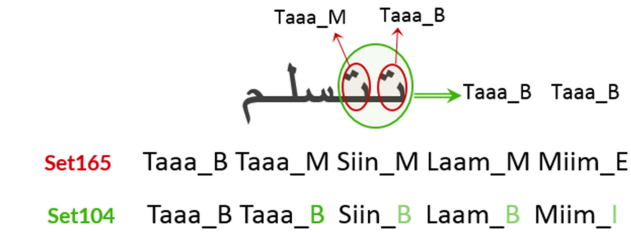

Fig. 8 *Sequence of models with proposed sets 165 and 104. 'B', 'M', 'E' and 'I', respectively, denote the letter positions Begin, Middle, End, and Isolate*

Table 1 Evaluation protocols and statistics of used dataset. Protocol 6 includes four sub-protocols: three channel-dependent protocols (6.1, 6.2, and 6.3) and one channel-free protocol (6.4)

Protocol	TV channel	Training-set			Test-set		
		#Lines	#Words	#Characters	#Lines	#Words	#Characters
P3	AlJazeeraHD	1,909	8,110	46,563	196	766	4,343
	France24	1,906	5,683	32,085	179	667	3,835
P6	Russia Today	2,127	13,462	78,936	250	1,483	8,749
	TunisiaNat1	2,001	9,338	54,809	189	706	4,087
	AllSD	6,034	28,483	165,830	618	2,856	16,671
P9	All	7,943	36,593	212,393	814	3,622	21,014

Fig. 9 *Typical textline images from the AcTiV-R dataset. From top to bottom: samples of the used TV channels: AljazeeraHD, TunisiaNat1, France24 Arabe, and RussiaToday Arabic*

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. 1*c*.

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:

$$\mathrm{CRR} = \frac{\#\mathrm{characters} - I - S - D}{\#\mathrm{characters}}, \tag{13}$$

$$\mathrm{WRR} = \frac{\#\mathrm{words\_correctly\_recognised}}{\#\mathrm{words}}, \tag{14}$$

$$\mathrm{LRR} = \frac{\#\mathrm{lines\_correctly\_recognised}}{\#\mathrm{lines}}. \tag{15}$$

6.3 Selection of the optimal network parameters

The optimal parameters for our proposed MDLSTM model are found by empirical analysis. Note that for these experiments we just pick out a small set of 2000 text images from AcTiV-R, in which 190 are used as a validation set. We first need to find the best size of hidden MDLSTM layers, which gives us the optimal performance. To do that, we fix the size of feedforward *tanh* layers to 6 and 20. As shown in Table 2, the suitable values of the MDLSTM size, which give us optimal results, are 2, 10 and 50 for the first, second and third hidden layers, respectively. Afterwards, we evaluate the impact of the feedforward size against the fixed optimal size of MDLSTM layers (2, 10 and 50). As a consequence, the best obtained size of feedforward layers is 6 and 20 for the first and second feedforward *tanh* layers, respectively, as presented in Table 3. The indicators observed during the fine-tuning of these parameters are the CRR and the average time per epoch. It is interesting to note that such results are not comparable with the system results obtained in the next subsection.

Once the architecture is fixed, we perform several experiments to find the best sizes of the input block and the hidden block (subsampling window). Therefore, the size of the input block is fixed to 1×4 for protocols 6.1 and 3, and to 2×4 for the remaining protocols. The hidden block sizes are fixed to 1×4 and 1×4 for all protocols.

It is worth noting that the training of this network is carried out with the back-propagation through time algorithm, and the steep-set optimiser is used with a learning rate of 10^{-4} and a momentum

Table 2 Impact of MDLSTM size against the fixed size of feedforward layer

MDLSTM size	CRR, %	LRR, %	Total epochs	Time per epoch, min
2, 10, 50	**96.26**	**68.26**	**128**	**7**
4, 20, 100	95.65	66.14	350	19
20, 60, 100	97.04	74.61	162	46
8, 30, 150	97.12	75.67	298	63

The bold values indicate the best values.

Table 3 Impact of feedforward layers size against the fixed size of MDLSTM layers

Feed-forward size	CRR, %	LRR, %	Total epochs	Time per epoch, min
6, 20	**96.26**	**68.26**	**128**	**7**
8, 30	96.46	71.43	347	13
12, 40	96.43	70.38	309	8
16, 80	97.09	75.7	204	17

The bold values indicate the best values.

Table 4 Results of the proposed recognition system on the AcTiV-R dataset: impact of polarity normalisation

	Without normalisation of text polarity			With normalisation of text polarity		
	CRR, %	WRR, %	LRR, %	CRR, %	WRR, %	LRR, %
P3	90.03	71.18	51.54	**92.2**	**74.13**	**53**
P6.1	89.1	70.49	51.4	**91.5**	**79.66**	**57**
P6.2	93.8	68.22	40.8	**93.33**	**68.9**	**43.6**
P6.3	94.3	80.77	62.44	**96.16**	**85.14**	**67.73**
P6.4	93.17	73	52.4	**94.1**	**81.23**	**57.3**
P9	73.4	58.34	31.9	**80.41**	**60.6**	**38.14**

Table 5 Final obtained results on the AcTiV-R dataset: impact of model sets choice

	Set165			Set104			Set72		
	CRR, (%)	WRR, (%)	LRR, (%)	CRR, (%)	WRR, (%)	LRR, (%)	CRR, (%)	WRR, (%)	LRR, (%)
P3	92.2	74.93	53.8	**94.62**	**83.11**	**64.29**	92.71	75.29	54
P6.1	91.5	79.66	57	**92.27**	**81.19**	**59.55**	91	75.18	52
P6.2	93.33	68.9	43.6	**94.1**	**73.67**	**49.27**	90.45	67.24	43.2
P6.3	96.16	85.14	67.73	**96.48**	**86.05**	**72.49**	93.87	82.39	63.6
P6.4	94.1	81.23	57.3	**95.82**	**83.4**	**63.27**	92.11	78.53	55.8
P9	80.41	60.64	38.14	**88**	**70.28**	**47.32**	80.77	59.11	36.4

The bold values indicate the best values.

value of 0.9. Training stops when the validation error shows no improvement in successive 20 epochs.

6.4 Results and discussion

6.4.1 Impact of the pre-processing step: To examine the impact of text polarity normalisation on the input greyscale images of each protocol, 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.

6.4.2 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* (ـٰـ) 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.

6.4.3 Error analysis: Fig. 10 depicts some typical misrecognised lines. 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)

Fig. 10 *Examples of some output errors picked out from experimental results. Errors are marked by red symbols*

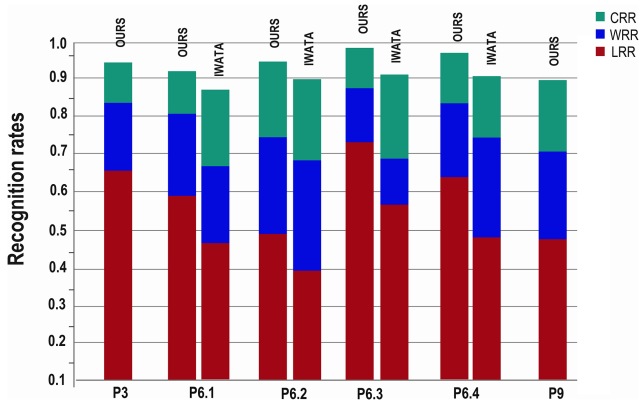

Fig. 11 *Comparison of our recognition system to Iwata's on the test-set of AcTiV-R*

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

Method	CRR, %	WRR, %	LRR, %
DBN-AE-BLSTM [16]	90.73	59.45	39.39
MLP-AE-BLSTM [16]	88.50	59.95	33.19
CNN-BLSTM [16]	94.36	71.26	55.03
HC-BLSTM [36]	85.44	52.13	—
ABBYY [36]	83.26	49.80	26.91
proposed system	**96.85**	**85.71**	**70.67**

The bold values indicate the best values.

R database. We have reported 96.5% as a character recognition rate and nearly 72.5% as a LRR for the SD resolution. The pre-processing step and model set choice have brought significant recognition improvement in terms of reduction in the line error rate. Our method has also outperformed the results of previous work on the ALIF dataset, more particularly those based on the combination of CNN and BLSTM [16, 36].

The interesting findings in this study have been the application of the MDLSTM network to video Arabic text with unknown font sizes and font families and the use of an efficient normalisation step as well as the analysis of the impact of model sets. Our results are achieved without using any dropout regularisation technique or language modelling. However, as a future work, we plan to integrate such methods to further improve the line recognition accuracy. Future work will also cover the development of an end-to-end recognition system which takes as an input the entire video sequence instead of text images.

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

[1] Mantas, J.: 'An overview of character recognition methodologies', *Pattern Recognit.*, 1986, **19**, (6), pp. 425–430

[2] Govindan, V.K., Shivaprasad, A.P.: 'Character recognition – a review', *Pattern Recognit.*, 1990, **23**, (7), pp. 671–683

[3] Dash, K.S., Puhan, N.B., Panda, G.: 'Handwritten numeral recognition using non-redundant stockwell transform and bio-inspired optimal zoning', *IET Image Process.*, 2015, **9**, (10), pp. 874–882

[4] Lorigo, L.M., Govindaraju, V.: 'Offline Arabic handwriting recognition: a survey', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2006, **28**, (5), pp. 712–724

[5] Srihari, S.N.: 'Recognition of handwritten and machine-printed text for postal address interpretation', *Pattern Recognit. Lett.*, 1993, **14**, (4), pp. 291–302

[6] Jayadevan, R., Kolhe, S.R., Patil, P.M., *et al.*: 'Automatic processing of handwritten bank cheque images: a survey', *Int. J. Doc. Anal. Recognit.*, 2012, **15**, (4), pp. 267–296

[7] Wang, F., Guo, Q., Lei, J., *et al.*: 'Convolutional recurrent neural networks with hidden Markov model bootstrap for scene text recognition', *IET Comput. Vis.*, 2017, **11**, (6), pp. 497–504

[8] Ye, Q., Doermann, D.: 'Text detection and recognition in imagery: a survey', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015, **37**, (7), pp. 1480–1500

[9] Yin, X.C., Zuo, Z.Y., Tian, S., *et al.*: 'Text detection, tracking and recognition in video: a comprehensive survey', *IEEE Trans. Image Process.*, 2016, **25**, (6), pp. 2752–2773

[10] Yu, C., Song, Y., Meng, Q., *et al.*: 'Text detection and recognition in natural scene with edge analysis', *IET Comput. Vis.*, 2015, **9**, (4), pp. 603–613

[11] Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., *et al.*: 'ICDAR 2015 competition on robust reading'. Proc. Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015, pp. 1156–1160

[12] Märgner, V., ElAbed, H.: '*Guide to OCR for Arabic scripts*' (Springer, London, 2012)

[13] Touj, S.M., Ben Amara, N.E., Amiri, H.: 'A hybrid approach for off-line Arabic handwriting recognition based on a planar hidden Markov modeling'. Proc. Int. Conf. on Document Analysis and Recognition, Curitiba, Paraná, Brazil, 2007, pp. 964–968

[14] Slimane, F., Zayene, O., Kanoun, S., *et al.*: 'New features for complex Arabic fonts in cascading recognition system'. Proc. Int. Conf. on Pattern Recognition, Tsukuba, Japan, 2012, pp. 738–741

[15] Graves, A.: 'Offline Arabic handwriting recognition with multidimensional recurrent neural networks', in '*Guide to OCR for Arabic scripts*' (Springer, London, 2012), pp. 297–313

[16] Yousfi, S., Berrani, S.A., Garcia, C.: 'Deep learning and recurrent connectionist-based approaches for Arabic text recognition in videos'. Proc. Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015, pp. 1026–1030

[17] Benhalima, M., Alimi, M.A., Vila, A.F., *et al.*: 'NF-SAVO: neuro-fuzzy system for Arabic video OCR', *Int. J. Adv. Comput. Sci. Appl.*, 2012, **14**, (1), pp. 128–136

[18] Schuster, M., Paliwal, K.K.: 'Bidirectional recurrent neural networks', *IEEE Trans. Signal Process.*, 1997, **45**, (11), pp. 2673–2681

[19] Senior, A.W., Robinson, A.J.: 'An off-line cursive handwriting recognition system', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1998, **20**, (3), pp. 309–321

[20] Robinson, A.J.: 'An application of recurrent nets to phone probability estimation', *IEEE Trans. Neural Netw.*, 1994, **5**, (2), pp. 298–305

[21] Graves, A., Liwicki, M., Fernández, S., *et al.*: 'A novel connectionist system for unconstrained handwriting recognition', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2009, **31**, (5), pp. 855–868

[22] Pham, V., Bluche, T., Kermorvant, C., *et al.*: 'Dropout improves recurrent neural networks for handwriting recognition'. Proc. Int. Conf. on Frontiers in Handwriting Recognition, Crete, Greece, 2014, pp. 285–290

[23] Messina, R., Louradour, J.: 'Segmentation-free handwritten Chinese text recognition with LSTM-RNN'. Proc. Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015, pp. 171–175

[24] Zhiwei, Z., Linlin, L., Lim, T.C.: 'Edge based binarization for video text images'. Proc. Int. Conf. on Pattern Recognition, Istanbul, Turkey, 2010, pp. 133–136

[25] Zhang, Z., Wang, W.: 'A novel approach for binarization of overlay text'. Proc. Systems, Man, and Cybernetics, Manchester, UK, 2013, pp. 4259–4264

[26] Hu, P., Wang, W., Lu, K.: 'A novel binarization approach for text in images'. Proc. Int. Conf. on Image Processing, Quebec City, Canada, 2015, pp. 956–960

[27] Roy, S., Shivakumara, P., Roy, P.P., *et al.*: 'Bayesian classifier for multi-oriented video text recognition system', *Expert Syst. Appl.*, 2015, **42**, pp. 5554–5566

[28] Zhai, C., Chen, Z., Li, J., *et al.*: 'Chinese image text recognition with BLSTM-CTC: a segmentation-free method'. Proc. Chinese Conf. on Pattern Recognition, Chengdu, China, 2016, pp. 525–536

[29] Su, B., Lu, S.: 'Accurate scene text recognition based on recurrent neural network'. Proc. Asian Conf. on Computer Vision, Singapore, 2014, pp. 35–48

[30] Jaderberg, M., Simonyan, K., Vedaldi, A., *et al.*: 'Synthetic data and artificial neural networks for natural scene text recognition'. NIPS Deep Learning Workshop, Montreal, Canada, 2014

[31] Elagouni, K., Garcia, C., Mamalet, F., *et al.*: 'Text recognition in videos using a recurrent connectionist approach'. Proc. Int. Conf. on Artificial Neural Networks, Lausanne, Switzerland, 2012, pp. 172–179

[32] Shi, B., Bai, X., Yao, C.: 'An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017, **39**, (11), pp. 2298–2304

[33] Iwata, S., Ohyama, W., Wakabayashi, T., *et al.*: 'Recognition and transition frame detection of Arabic news captions for video retrieval'. Proc. Int. Conf. on Pattern Recognition, Cancun, Mexico, 2016, pp. 4005–4010

[34] Khémiri, A., Kacem, A., Belad, A., *et al.*: 'Arabic handwritten words off-line recognition based on HMMs and DBNs'. Proc. Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015, pp. 51–55

[35] Abandah, G.A., Jamour, F.T., Qaralleh, E.A.: 'Recognizing handwritten Arabic words using grapheme segmentation and recurrent neural networks', *Int. J. Doc. Anal. Recognit.*, 2014, **17**, (3), pp. 275–291

[36] Yousfi, S., Berrani, S.A., Garcia, C.: 'ALIF: a dataset for arabic embedded text recognition in TV broadcast'. Proc. Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015, pp. 1221–1225

[37] Naz, S., Umar, A.I., Ahmad, R., *et al.*: 'Urdu Nastaliq text recognition system based on multi-dimensional recurrent neural network and statistical features', *Neural Comput. Appl.*, 2017, **28**, (2), pp. 219–231

[38] Chherawala, Y., Roy, P.P., Cheriet, M.: 'Feature set evaluation for offline handwriting recognition systems: application to the recurrent neural network model', *IEEE Trans. Cybern.*, 2016, **46**, (12), pp. 2825–2836

[39] Yousefi, M.R., Soheili, M.R., Breuel, T.M., *et al.*: 'A comparison of 1D and 2D LSTM architectures for the recognition of handwritten Arabic'. Document Recognition and Retrieval, San Francisco, USA, 2015

[40] Lee, S.-W., Kim, Y.-J.: 'A new type of recurrent neural network for handwritten character recognition'. Proc. Int. Conf. on Document Analysis and Recognition, Montreal, Canada, 1995, pp. 38–41

[41] Graves, A., Fernandez, S., Schmidhuber, J.: 'Multi-dimensional recurrent neural networks'. Proc. Int. Conf. on Artificial Neural Networks, Porto, Portugal, 2007, pp. 549–558

[42] Graves, A.: '*Supervised sequence labelling with recurrent neural networks*', (Springer, Berlin, Heidelberg, 2012)

[43] Bengio, Y., Simard, P., Frasconi, P.: 'Learning long-term dependencies with gradient descent is difficult', *IEEE Trans. Neural Netw.*, 1994, **5**, (2), pp. 157–166

[44] Hochreiter, S., Schmidhuber, J.: 'Long short-term memory', *Neural Comput.*, 1997, **9**, (8), pp. 1735–1780

[45] Austin, S., Schwartz, R., Placeway, P.: 'The forward-backward search algorithm'. Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Toronto, Canada, 1991, pp. 697–700

[46] Slimane, F., Ingold, R., Alimi, M.A., *et al.*: 'Duration models for arabic text recognition using hidden Markov models'. Proc. Computational Intelligence for Modelling Control & Automation, Vienna, Austria, 2008, pp. 838–843

[47] Zayene, O., Hennebert, J., Touj, S.M., *et al.*: 'A dataset for arabic text detection, tracking and recognition in news videos – AcTiV'. Proc. Int. Conf. on Document Analysis and Recognition, Nancy, France, 2015, pp. 996–1000

[48] Zayene, O., Hajjej, N., Touj, S.M., *et al.*: 'ICPR2016 contest on arabic text detection and recognition in video frames-AcTiVComp'. Proc. Int. Conf. on Pattern Recognition, Cancun, Mexico, 2016, pp. 187–191

[49] Frinken, V., Zamora-Martnez, F., Espana-Boquera, S., *et al.*: 'Long-short term memory neural networks language modeling for handwriting recognition'. Proc. Int. Conf. on Pattern Recognition, Tsukuba, Japan, 2012, pp. 701–704