Multimedia Tools and Applications (2019) 78:26453–26481
https://doi.org/10.1007/s11042-019-07855-z

Text recognition in document images obtained by a smartphone based on deep convolutional and recurrent neural network

Hassan El Bahi[1,2] · **Abdelkarim Zatni**[1]

Received: 31 August 2018 / Revised: 16 March 2019 / Accepted: 31 May 2019 /
Published online: 11 June 2019

Abstract
Automatic text recognition in document images is an important task in many real-world applications. Several systems have been proposed to accomplish this task. However, a little attention has been given to document images obtained by mobile phones. To meet this need, we propose a new system that integrates preprocessing, features extraction and classification in order to recognize text contained in the document images acquired by a smartphone. The preprocessing phase is applied to locate the text region, and then segment that region into text line images. In the second phase, a sliding window divides the text-line image into a sequence of frames; afterwards a deep convolutional neural network (CNN) model is used to extract features from each frame. Finally, an architecture that combines the bidirectional recurrent neural network (RNN), the gated recurrent units (GRU) block and the connectionist temporal classification (CTC) layer is explored to ensure the classification phase. The proposed system has been tested on the ICDAR2015 Smartphone document OCR dataset and the experimental results show that the proposed system is capable to achieve promising recognition rates.

Keywords Text recognition · Document image · Smartphone · Convolutional neural network · Recurrent neural network

1 Introduction

In recent years, technological development has radically changed our relationship with audiovisual information. Images and videos have taken an important place in the flow of information that we receive every day. The way we treat this flow of information would certainly affect our daily lives. The text documents (books, printed articles, patent documents,

✉ Hassan El Bahi
hassan.elbahi@edu.uiz.ac.ma

[1] Laboratory of Metrology and Information Processing, Ibnou Zohr University, Agadir, Morocco

[2] Laboratory of Applied Mathematics and Computer Science, Cadi Ayyad University, Marrakech, Morocco

receipts, magazine pages, etc.) are one of the forms of presentation and organization of information. Lately, taking photos of text documents in place of digitizing them with a scanner has become more and more common due to the popularity and the rapid development of smartphones. Several works have been proposed for the detection and recognition of the text contained document images. Nevertheless, most systems, generally called Optical Character Recognition (OCR) systems, have been dedicated to scanned text documents [61, 64, 65, 68] although the text document images acquired by mobile devices that represent a large part of multimedia content. The processing and recognition of this last category of document images is not a trivial task and represents a major challenge [40].

In general, the processing of an image obtained by a smartphone is affected by several factors, such as the perspective distortion that can takes place when the text image plan to capture is not parallel to the smartphone's camera plane. The result of this distortion is that the characters located farther appear smaller and the hypothesis of parallel lines of the edges of the document page no longer fits to those in the captured image. We can also point out that mobile devices have much less control over the lighting conditions of the acquisition environment. The variation in illumination is common, due to the physical environment such as shadows or reflective surfaces and lack of controlled lighting. Complex background is also another problem that can constantly face when the scene to be captured is larger than the text region. Blur distortion is another problem that appears on document images when

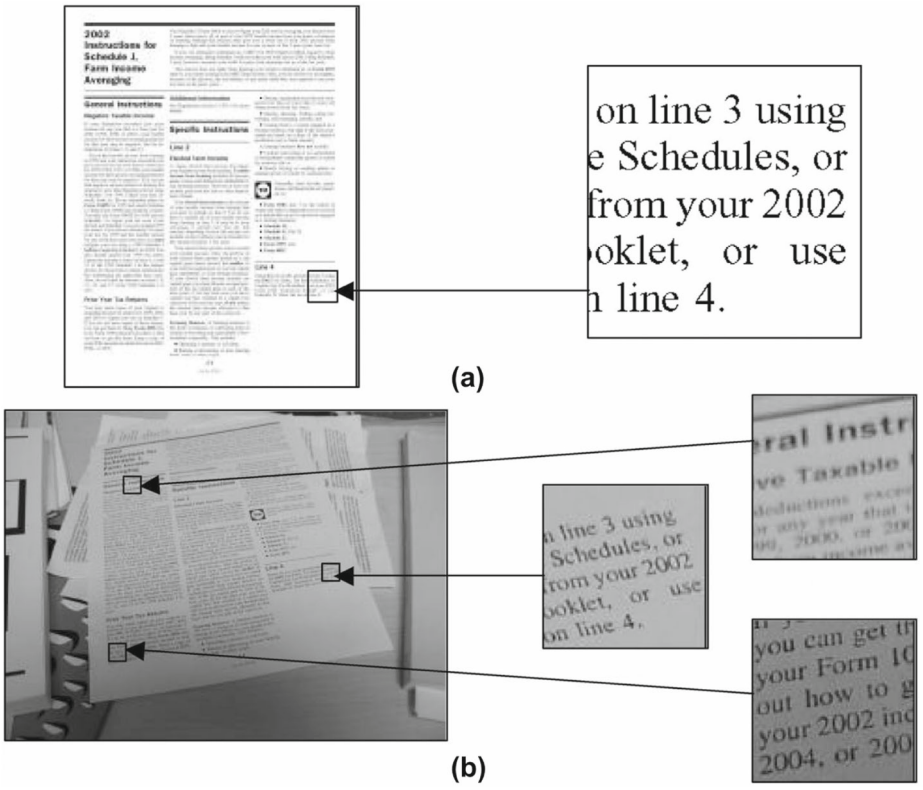


Fig. 1 Challenges of document images acquired by a mobile terminal [30]. **a** Document scanned using a scanner. **b**: The same document captured by a mobile terminal

the smartphone's camera is focused on the background rather than the text document, or may be caused by the movement of the Smartphone's camera during acquisition. The processing of images obtained by smartphone is also affected by the variety of text properties from one document text to another, such as: variety of sizes, fonts, styles, colors. Figure 1 presents some examples of the problems posed after the acquisition of a document image by a mobile device. All the aforementioned challenges have formed our motivation to propose a system for text recognition in document images obtained by mobile terminals.

Text recognition can be defined as the ability of a smartphone or a computer to transform text data contained in an image into its equivalent symbolic representation, in the form of ASCII text. Typically, a text recognition system comprises three main phases: preprocessing, features extraction and classification [43]. The preprocessing step is generally applied to improve and enhance the quality of the document image for subsequent appropriate analysis. This step can itself be divided into smaller steps such as: text detection, perspective correction, segmentation, skew correction, normalization and so on [39]. In the preprocessing phase, a detection method takes a gray scale or color image as input and outputs an image with a rectangle surrounding the text region. The methods proposed in the literature for this task can be classified into three categories [25]: connected component-based methods [22, 62, 63], texture-based methods [16] and gradient-based methods [49, 58]. The perspective correction step makes it possible to transform the geometry of the image into a perspective different from that which was originally captured. Many works have been done to accomplish this step. kim and al [26] use a discrete representation of text-lines and text-blocks to correct image document perspectives. Castro et al. [9] propose a method that first consists in detecting the text region in the image, and therefore the perspective correction is performed using a transformation matrix. Liu et al. [31] present an approach that is based on the segmentation of text into individual lines, and then perspective correction is performed at each line of text [46]. With regard to segmentation, several approaches are reported in the literature to ensure this step. They are categorized into three strategies: top-down, bottom-up and hybrid [14]. Top-down strategies like projection profile [4], filtering techniques [6, 55] and Hough transform [46] take as input the entire image of text and attempt to divide it into different text-lines images. In terms of the bottom-up strategies, they start from a small-scale level of an image (i.e. pixels), and afterwards they use some technique like: clustering [32, 69], function analysis [2] or active contours [42], in order to locate each text line area. While other works [51, 56] are trying to combine the two strategies for segment the text image into lines. The purpose of the next step is to detect and correct the skew of each text-line image. Many methods have been developed to this end [15, 36]. Generally, these methods are based on the following idea: first, they try to detect the angle of inclination, and then the correction is performed using rotation by this angle. After skew correction, size normalization is an important task which aims to minimize the variations between the shapes of the characters and words in the different document images of a database. In the case of text line images, normalization consists in forcing each line image to take the same height while maintaining the width/height ratio of the images [23].

Extraction of representative and essential features from an input image is the main key to improving the performance of a recognition system. According to the literature [33], feature extraction methods can be divided into two groups: hand-crafted feature-based methods and automatic learning feature-based methods. In the case of the first group, we can mention the direct use of pixel intensity values as features [20, 35], number and positions of black pixels [5] and statistical features [24, 41]. In addition, we can add other methods like: the use of histogram of oriented gradients (HOG) as a descriptor [53], projections of oriented

gradients [44], principal component analysis (PCA) [27], etc. However, the traditional feature extraction methods are very limited as they require prior knowledge about information on the position and relevance of features, and they are more sensitive to the variation of font sizes, styles, colors and non-uniform lighting condition of document text [33]. In the last few years, the automatic learning feature methods, particularly the deep convolutional neural network (CNN), has shown better performance compared to hand-crafted features methods in various recognition problems such as: image classification [60, 67], cancer detection [59], age estimation [10], Pedestrian Detection [29], etc. The deep convolutional neural network is an alternative that does not suffer from the disadvantages already mentioned in the previous group of methods. Their deep architecture allowed them to be able to learn automatically the most relevant features, which makes it one of the most robust systems in terms of translation, scaling and distortions. We can cite the adaptation of the CNN model as a features extraction method in several text recognition systems such as: scene text recognition [50], offline handwriting recognition [54], video text recognition [66], and historical handwriting recognition [17]. However, the only drawback of automatic learning feature-based methods versus hand-crafted feature-based methods is that they require more time in the training phase to find the most relevant features [57].

The feature vector extracted in the last phase is passed as input to an already trained classifier that predicts its class. Numerous approaches have been used to classify text in images. They are categorized into two groups: segmentation-based approach and segmentation-free approach. The approaches in the first group perform explicit segmentation of text-line image into characters before employing a classifier like MLP or SVM [57] to recognize the class of each individual character. However, the performance of these approaches is strongly related to the results of decomposition of text-line into characters. Each segmentation error directly decreases the recognition rate of the entire system. The segmentation-free approaches present an alternative to overcome this limitation. They take as input the whole text-line image as a sequence of images prepared by a sliding window, which allows recognizing the text image without performing explicit segmentation. As a result, classification models based on this group of approaches are the most used especially in the case where separations between characters are difficult to determine, such as: handwriting, complex background, overlapping characters, and so on. Furthermore, these approaches make it possible to use the contextual information, which means that the output is calculated at each time step according to the past and future contexts of a character or a word [20]. The two most frequently used models are the Hidden Markov Model (HMM) and the Recurrent Neural Network (RNN). The HMM model is used in many text recognition systems, for example: English texts [45], Arabic text and [1]. In recent years, with the fast advancement of computers and the use of GPU for calculation, the RNN model demonstrates better performance than the HMM model in the task of text classification [18]. Numerous recent works have taken advantage of the power of the RNN, for example and without limitation, recognition of French text [36], recognition of English text [50], recognition of Arabic text [34].

The main contribution of this work is to address the problem of text recognition in images obtained by a smartphone. This is realized by the proposal of a new system that consists of three main parts. In the first one, a preprocessing operation is applied in order to detect text region in document image, and then segment this region into individual text-line images. This phase takes into consideration the problems mentioned above, often encountered during the acquisition of images by mobile. Inspired by the great success achieved thanks to the automatic learning feature-based methods [50], a deep convolutional neural network CNN model is used in the second part to extract a sequence of feature vectors form each text-line image. In this way, the system will be able to automatically extract the features that best

discriminate between the different classes of characters, rather than using a hand-crafted feature-based technique. The last part of the system is dedicated to the classification phase where we use a bidirectional recurrent neural network (BRNN) combined with the gated recurrent units (GRU) block [11] and the Connectionist temporal classification (CTC) layer [18]. The BRNN-CTC architecture takes as input the sequence of feature vectors and produces a sequence of characters representing the text contained in the text-line image. BRNN offers the possibility to use past and future context information in the calculation, which is useful and indispensable in the recognition of the image text sequences. Using a CTC layer eliminates the need to segment the text line image into images of individual characters. It takes as input the text line image and gives as an output the corresponding text transcription without having to perform an explicit segmentation. The proposed system is evaluated on the ICDAR2015 Smartphone document OCR dataset [7]. The main contributions of the proposed system can be summarized as follows:

– We implement and compare different CNNs models to extract features from text-line images according to different widths of a sliding window.

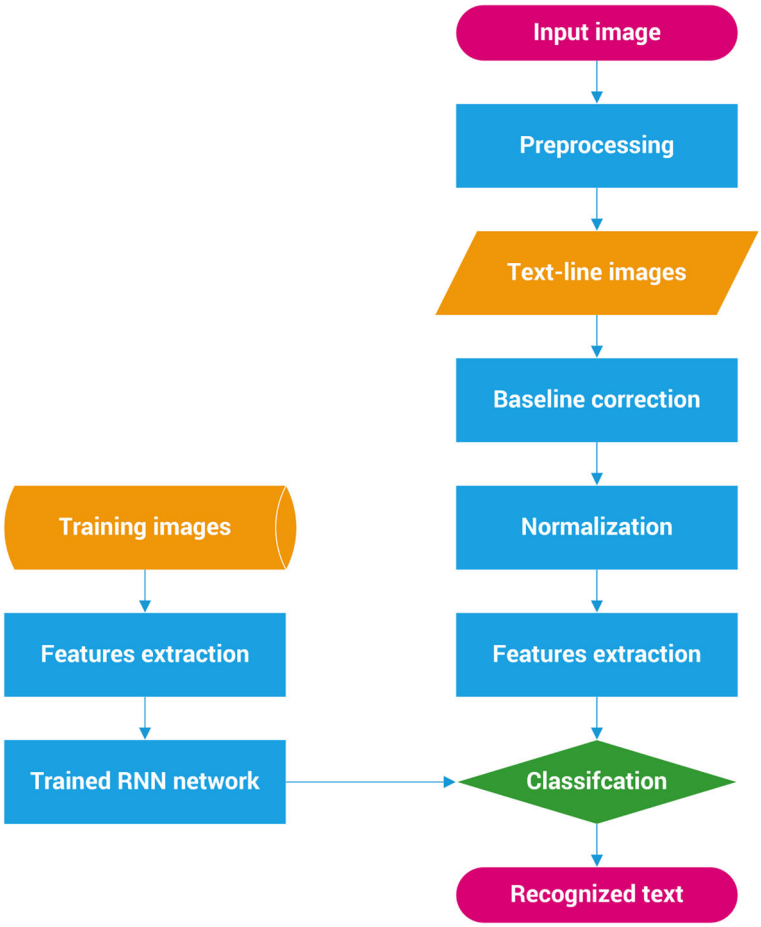


Fig. 2 Flowchart of the proposed text recognition system

– We explore various combination between a RNN or BRNN with LSTM [21] or GRU [11] memory block to classify the sequence of feature vectors that extracted by CNN model. As a results, the BRNN-GRU architecture outperforms the BRNN-LSTM architecture and provides a better recognition rate.
– The experimental tests of the proposed system on the ICDAR2015 Smartphone document OCR dataset [7] shows promising results compared to the state-of-the-art systems.

The remainder of this paper is structured as follows: Section 2 describes the details of each step of the proposed text recognition system, whereas Section 3 presents the experimental results. Finally, Section 4 presents conclusion and discusses future works.

2 Proposed system

As shown in the Fig. 2, the general architecture of the proposed system includes five major steps: preprocessing, baseline correction, normalization, feature extraction and classification. This section details each step of the proposed system.

2.1 Preprocessing

The preprocessing phase is essential for detecting text, eliminating noise and segmenting text into lines of text. Figure 3 shows all the operations used during the preprocessing phase. These operations will be explained in detail in the following parts.

2.1.1 Text detection

In this part, we present the method used for the detection of the text region. The method consists of several steps, the first of which consists of converting the original image into a grayscale image (see Fig. 4b). The second step is to detect all the contours of the image, for this reason we use Canny's algorithm [8]. Figure 4c shows the result of this algorithm.

The goal of the next step is to help distinguish the text region from the background of the image. For this reason, we apply the dilation technique. The main idea of this technique is to connect and enlarge groups of nearby pixels in an image. In our case, this will merge the characters and fill the spaces between the words to form bands corresponding to each line of text (see Fig. 4d). At this point, the lines of text are clearly visible, but there are still black areas between the lines of text and the outlines of the text region. Therefore, the next step is to fill these areas with white pixels. This is done as follows: If a black region is surrounded by white outlines, the pixels in that region will be transformed into white pixels. Figure 4e shows the regions filled after the application of this operation.

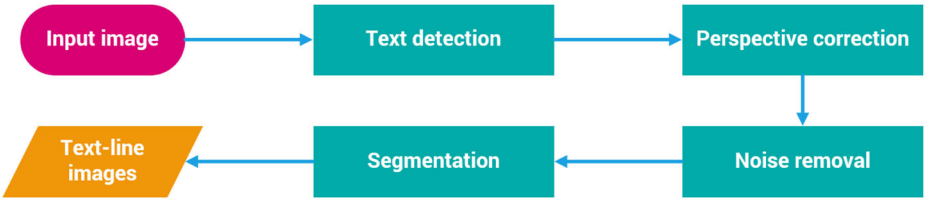


Fig. 3 Preprocessing operations

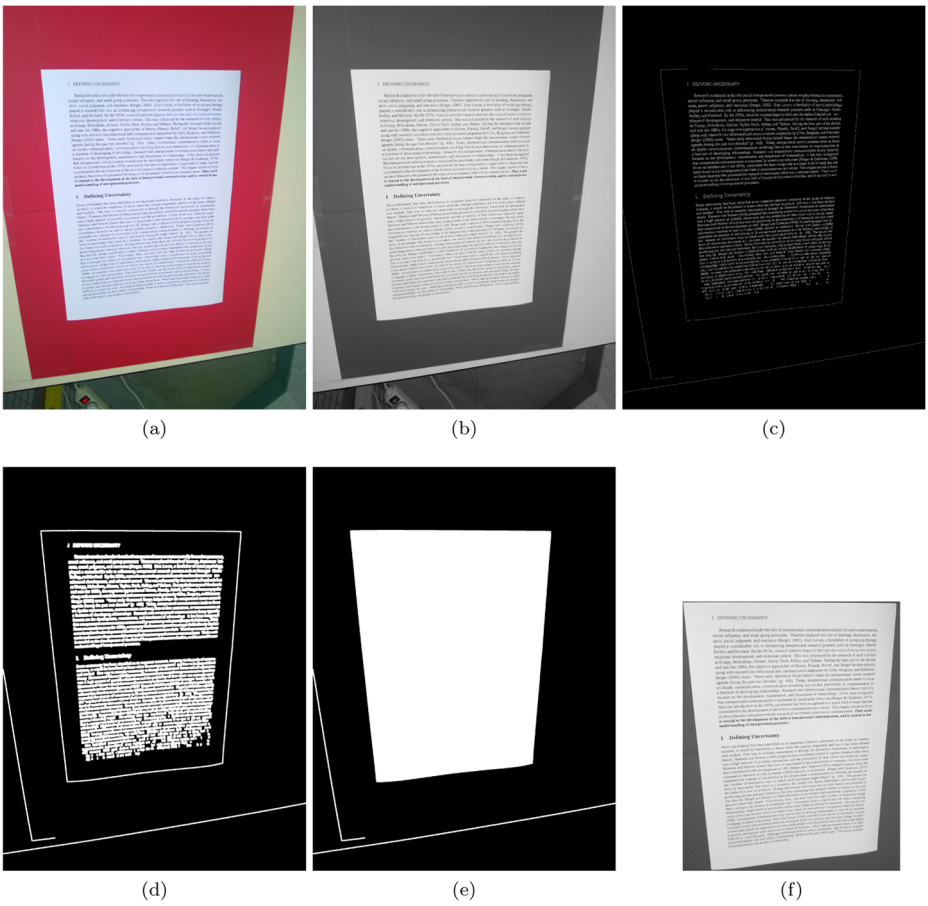
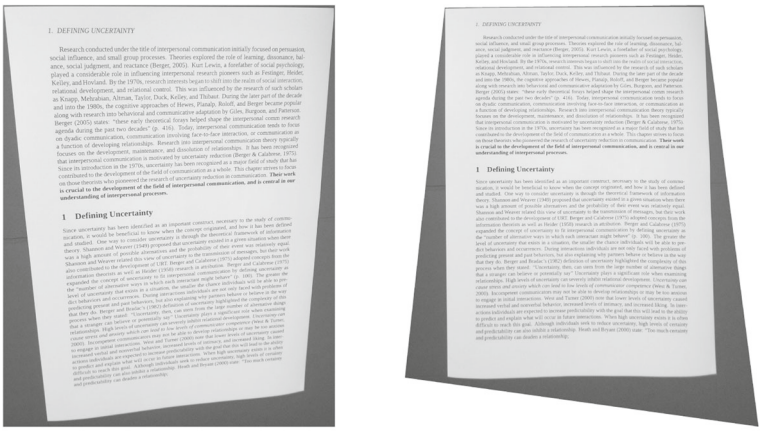


Fig. 4 The steps for detecting the text region. **a**: Initial image. **b**: Gray scale image. **c**: Contour detection. **d**: The dilated image. **e**: The filled image. **f**: Text region

The last step is the determination of the text region from the regions obtained in the previous step. To achieve this result, we rely on the fact that the document page is the main object in all images. Thus, we count the number of pixels in each region of the image, and we choose the region that contains the largest number of pixels. Figure 4f shows the result obtained after this last step.

2.1.2 Perspective correction

During the acquisition of an image, effect like perspective distortion is inevitable. This effect can damage the image and decrease the performance of the recognition. Therefore, perspective correction is a mandatory task to improve and reduce the complexity of an image. To perform this task, we use our previous work [12]. The approach proposed is designed to quadrilateral forms, especially the form of a document page. It is divided into two phases involving the detection of the corners of the document page, and hence the use of



(a) (b)

Fig. 5 The perspective correction of a document image. **a** Initial image. **b** Result

bilinear interpolation to correct the perspectives. Figure 5 shows an example of a document image before and after perspective correction.

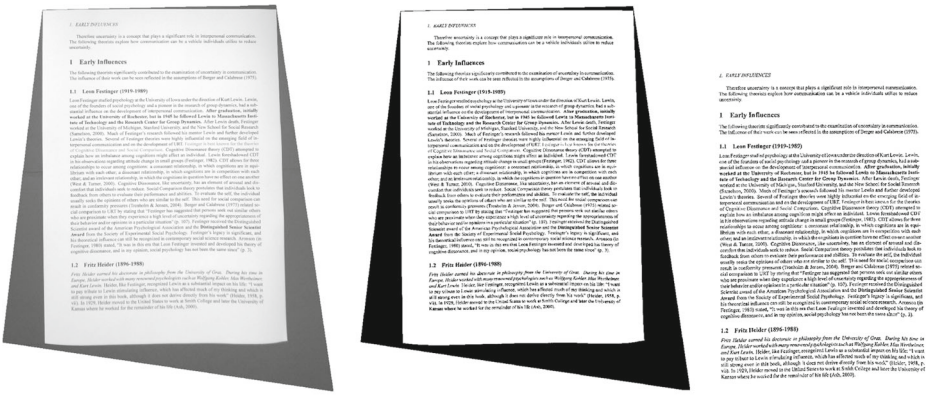
2.1.3 Noise removal

After the phases of text extraction and perspective correction, the noise appears on certain areas of the image, especially at the edge of the text page.This can cause instability and errors during the segmentation and classification phases. In this work, we use two operations to solve this problem: binarization and marginal noise suppression. The operation of binarization is provided by using Sauvola algorithm. Then the projection histogram is calculated to detect the noise of the horizontal (top and bottom) and vertical (left and right) margins. In the case of the noise of the horizontal margin, we plot the horizontal projection with the calculation of the number of black pixels along each line of the binary image. From the obtained curve, it is possible to determine the location of the upper and lower zone of the marginal noise according to the location of the first and last peak respectively. Then, we look for the location where the top horizontal cut will be done, according to several experiments, we opted to choose the location between the first and the second peak. In the same way, we chose the location between the last and the penultimate peak for horizontal cutting at the bottom. To remove noise from the vertical margin, we use the same way but this time vertically by counting the number of black pixels along each column. Figure 6 illustrates the results of binarization and marginal noise suppression.

2.1.4 Segmentation

In this part we are interested in the task of segmenting the text of the image into individual lines images. The main problem encountered in this step is the inclination of the lines of text with respect to the horizontal axis of the image, as shown in Fig. 7a.

To solve this problem, we use an algorithm that based on the idea that an image of the text can be grouped as related components separated by white spaces along each line of text [3]. The proposed algorithm works as follows: the input of the algorithm is the document image, a threshold S which designates the minimum size of a connected component and the



(a) (b) (c)

Fig. 6 Removing the marginal noise. **a** Initial image. **b** Binarized image. **c** Result

parameter T which corresponds at the average distance which makes it possible to know if two connected components belong to the same line or not. Then, the morphological dilation is applied to extend and reinforce the connected components of text. his is very interesting because it allows to group the words in each line of text (see Fig. 7b). After the previous operation, all the CCs related components of the image are extracted (see Fig. 7c). These CCs are grouped into large R components and small SR components (The size of a CC is determined by the number of pixels it contains). afterwards, all the components of the R group are sorted in decreasing order from top to bottom, and the SR group is eliminated initially, since the small components are usually far from the main body of the words or characters, making the segmentation process more difficult to accomplish.

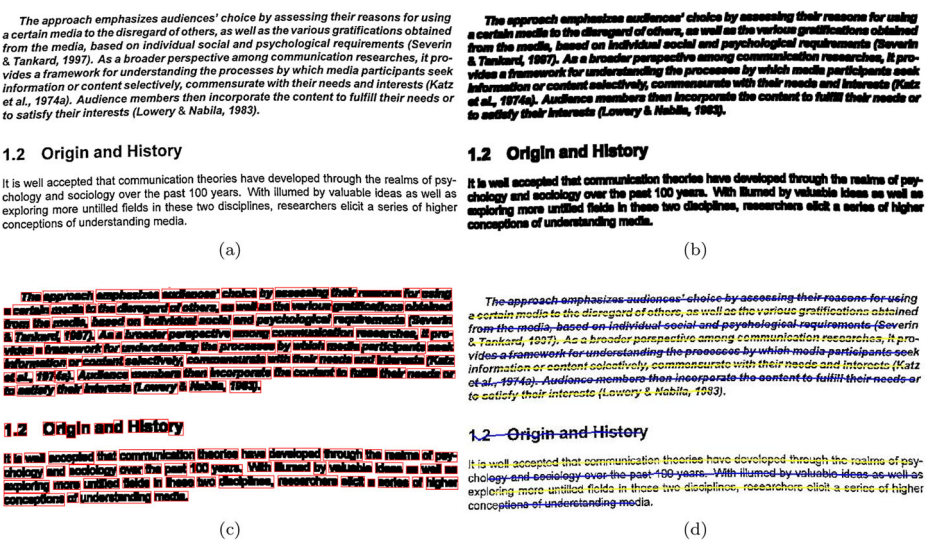


Fig. 7 Segmentation of a text image. **a** Initial image. **b** Dilated image. **c** Extraction of connected components. **d** The final result

In the second part of the algorithm, we select the first connected component $R1$ which will be located in the highest part of the image, and we assign it to line 1. After that, we identify the nearest $CC\ R2$ next to $CC\ R1$. This is done as follows: assuming that (x_1, y_1) and (x_2, y_2) are the coordinates of the centroids of $CC\ R1$ and $CC\ R2$ respectively. $R2$ belongs to the same line of $R1$ only if the condition $|y_2 - y_1| < T$ is satisfied, where T is the distance between the two $CCs\ R1$ and $R1$. The same process will be repeated until all the connected components are assigned to lines of text. In the last part of the algorithm, we affect each small component SRi to the line Li at which it has the shortest distance. Figure 7d illustrates the result obtained using the proposed algorithm.

2.2 Baseline correction

The text-line inclination problem is often appears after the segmentation phase, (see Fig. 8a below). This problem presents a difficulty in the classification phase in which we use a sliding window to extract features. An inclined writing can lead to an overlap and instability of characters and words within the sliding window. As a result, correcting the angle of inclination of each line of text is a necessary step before moving to the next steps.

The inclination correction of a line of text (also called skew correction), consists in straightening and putting horizontally the line of the inclined writing. In this work, we adopted the technique described in the reference [37], to correct the angle of inclination. This technique converts the text line image to an image composed of black block strips having a rectangle shape all along the text line, and then it uses these blocks to identify the candidate points to plot the approximation of the baseline (see Fig. 8b). Next, a fourth degree polynomial is used to draw the baseline approximately by fitting a curve through the extracted candidate points. Figure 8c and d below illustrate the baseline plot result with the curve fit operation.

After the baseline estimate, the average inclination angle for each block is calculated using the location of the candidate points (pixels) and the location of the baseline intersection points with the block. Then each block presented in the text line image is rotated by the

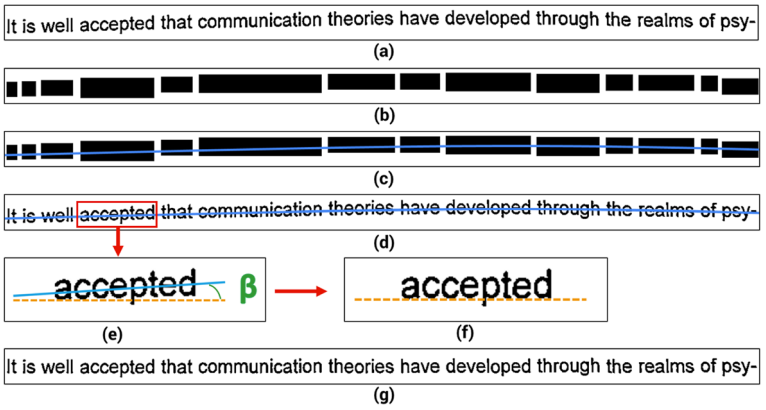


Fig. 8 Baseline correction of a text-line image. **a**: Initial image. **b**: Painted strips image. **c**: Baseline estimate. **d**: Baseline tracing on the initial image. **e** Estimation of the angle of inclination of the word "accepted". **f**: Correction of inclination of the word "accepted". **g**: Image results after correction of the angle of inclination

corresponding inclination angle in order to horizontally set the baseline of the image (see Fig. 8e and f). Figure 8g shows the result of the technique used for the inclination angle correction of the text line image.

2.3 Normalization

The size of a text image may vary from one writer to another, from one font to another and within the same font after enlarging or decreasing in size, which can cause instability in performance of recognition. Therefore, the normalization is an important task that aims to minimize variations between the shapes of characters and words in different images. In the case of text line images, the normalization phase consists in forcing the images of lines with an arbitrary dimension to have the same height. The width of the resized image is determined relative to the height of the original image; this operation is performed without affecting the height-to-width ratio of the text line image. We have observed that the normalization of all text-line images to a fixed height (32 pixels) does not have much influence on system performance. The results obtained after normalization of the size of some text-line images are shown in Fig. 9.

2.4 Features extraction

Features extraction is the process of taking a dataset of images and building explanatory variables (Features). After that, these features will be employed in order to train a machine learning or a deep learning model for a classification problem. In particular, the features extraction step aims to obtain the most relevant features of a text line image, while avoiding the loss of important and significant information of the image, and subsequently represents this information as a of one-dimensional vector. Inspired by the system proposed in [50], we adapt the CNN model in the features extraction step in order to build a text-line classifier. The model will be able to automatically extract the features that best discriminate between the different classes of characters; this is done using a large training set of text-line images. In general, a deep CNN architecture consists of convolution layers, nonlinear layer (The ReLU layer), pooling, and fully connected layers [28]. The rest of this section gives the concept of each layer and the adaptation methodology of CNN in our work.

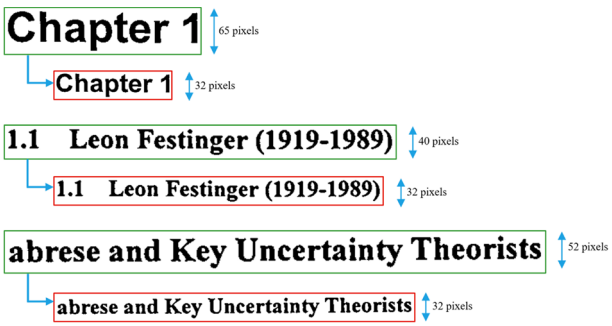


Fig. 9 Normalization of text-line images size to a height of 32 pixels while maintaining the aspect ratio

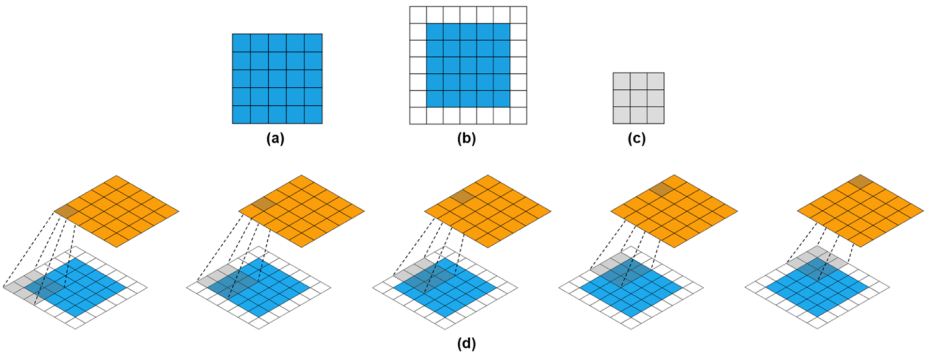


Fig. 10 Example of the convolution operation. **a**: Initial image. **b**: Image with padding of 1 pixel. **c**: Filter. **d**: The convolution between the filter I and each part of the image with a stride of 1 pixel

2.4.1 Convolutional layers

The convolution layer is the key component of CNN, and is always the first layer. Its purpose is to locate the presence of a set of features in the image received as input. This is thanks to a succession of filters also called kernels, transforming the initial image into a set of images called feature maps (or activation maps). In fact, each filter F_j of size (W_{F_j}, H_{F_j}) is started from the top-left corner and traverses the entire input image $I(W_I, H_I)$. In every location an element wise multiplication is performed between the image patch and the filter. The sum of this multiplication is then constituted one of the pixels of the feature map. In addition to the number and the size of filters, there are two common hyperparameters for convolution layer: stride and padding. Stride indicates the number of pixels with which the filter shifts horizontally or vertically on the input image. For example, a stride of one pixel means moving the filter one pixel at each time step to compute the next convolution output. The padding (Zero-padding) indicates how many rows and columns of 0 are added around the input image, which allows to control the size of the output. It is sometimes desirable to keep the same size as that of the input image. For better understanding the convolution layer, Figure 10 illustrates an example of the convolution operation. In the example, an image I of a size of (5, 5) (see Fig. 10a) is used with a padding of 1 pixel (see Fig. 10b). The image is convoluted with a filter F of a size of (3, 3) (see Fig. 10c). Figure (see Fig. 10d) shows the convolution product calculation between the filter I and each portion of the image I. The filter traverses the entire image with a stride of 1 pixel. The result at each step is one of the elements of the feature map matrix.

2.4.2 Nonlinear layer

After each convolution layer, a rectified Linear Units (ReLU) layer is added. It goal is to introduce and increases non-linearity in CNN by applying an activation function defined as follows: $f(x) = max(0, x)$ for each pixel in the feature map. The concept of this function is fairly simple: every time there is a negative value in a pixel, it is replaced by a 0. ReLU has been widely used due to its ability to learn fast and works better compared to traditional non-linear functions (tanh or sigmoidal) [38]. An example of using the function ReLU is shown in Fig. 11.

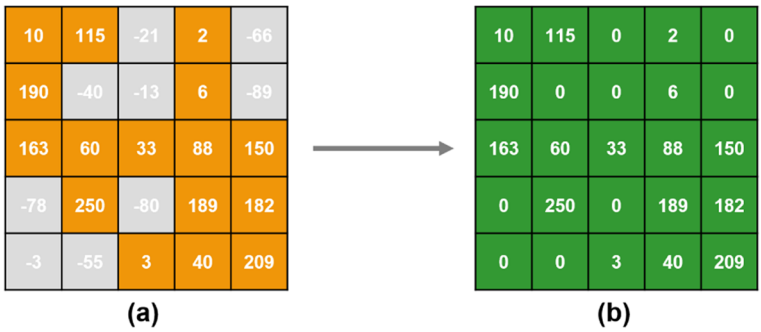


Fig. 11 Functionality of the Rectified Linear Units layer. **a**: Feature map. **b**: Rectified feature Map

2.4.3 Max-pooling layers

Another very powerful tool used by CNNs is called Pooling. It's a layer often placed between two convolution layers designed to take a large image and reduce its size while preserving the most important information it contains. This serves two purposes; first it makes the CNN less sensitive to the position of features and robust against noise and distortion. Second, it improves the efficiency of the CNN and avoids overfitting [28]. There are different manners to do pooling, but in practice the most effective is max pooling [47]. It consists of taking a kernel and passing it over the rectified feature map as in convolution, but this time we take the maximum value in every image patch that the kernel convolves around. In practice, a kernel of (2×2) pixels is often used with a value of 2 pixels for the stride and without any padding. Figure 12 illustrates an example of the Max Pooling application on the rectified feature map that has been obtained in the previous layer.

2.4.4 Fully connected layerss

After several layers of convolution and max-pooling, the last layer is called fully connected layer. Each neuron in this layer is connected to all the neurons of the previous layer. This

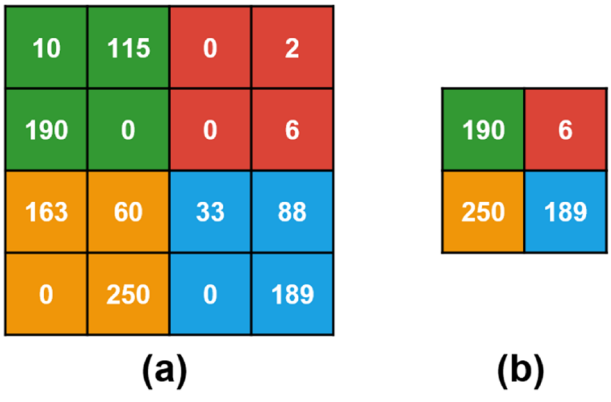


Fig. 12 Example of a max-pooling operation with a kernel of (2×2) pixels, a stride of 2 pixels and without any padding. **a**: Rectified feature Map. **b**: Result

Fig. 13 Representation of the sliding window approach and the CNN architecture that includes five convolutional layers and five max pooling layers

layer gathers all the information processed separately in the previous layers, and provides the final result (vector of probabilities) using a sigmoid or softmax activation function. The size of the result vector is equal to the number of classes of classification.

2.4.5 Methodology adaptation

In this paper, we directly extract the features from the text-line image based on deep convolutional neural networks. Before that, a sliding window approach vertically scans the text line image from left to right using a fixed-length window; this will allow dividing the image into sub images, called frames. The use of this approach requires adjusting two parameters: the shift s and the size of the windows ($W \times H$) (see Fig. 13). In fact, we use a shift of 4 pixels and the height H of the window is corresponding to the height of the image (32 pixels). After decomposing the images into a set of overlapping frames, the features of each frame ($2D$) are extracted using a CNN model in order to form the corresponding feature vector ($1D$). A visualization of the sliding window approach and the CNN model is illustrated in Fig. 13.

In this work, we implement and compare three different CNNs models applied to a different width. These models are built according to the size of three frames: 4×32, 8×32 and 16×32, and are referred to hereinafter as $CNN_{4\times32}$, $CNN_{8\times32}$ and $CNN_{16\times32}$. Table 1 presents the architecture of each of these modules. The three CNNs are composed of 5 convolutional layers and 5 max pooling layers (the fully connected layer is removed). Every convolution filter has a filter size of 3×3 pixels with stride of 1×1 pixels and a zero padding of 1×1 pixels. The number of feature maps in these layers is 64, 128, 256, 512 and 512. To introduce non-linearity, a ReLU activation function is applied after each convolution layer. A rectangular kernel of 1×2 pixels is applied at the $1st$, $2nd$ and $3rd$ max-pooling layers for the $CNN_{4\times32}$ model, at the $1st$ and $2nd$ max-pooling layer for the $CNN_{8\times32}$ model and only at the $1st$ max-pooling layer for the $CNN_{16\times32}$ model. A square kernel of 2×2 pixels is used for the remaining max-pooling layers. All the max-pooling layers used a stride of

Table 1 The structure of each CNN model. The output, maps, F, K, S and P are respectively: the size of feature map, the feature maps number, filter, kernel, stride and padding

Layer	CNN models		
	$CNN_{4\times32}$	$CNN_{8\times32}$	$CNN_{16\times32}$
Convolution 1	Output: 4×32 maps: 64 S: 1×1 P: 1×1 F: 3×3	Output: 8×32 maps: 64 S: 1×1 P: 1×1 F: 3×3	Output: 16×32 maps: 64 S: 1×1 P: 1×1 F: 3×3
Max-pooling 1	Output: 4×16 maps: 64 S: 1×2 K: 1×2	Output: 8×16 maps: 64 S: 1×2 K: 1×2	Output: 16×16 maps: 64 S: 1×2 K: 1×2
Convolution 2	Output: 4×16 maps: 128 S: 1×1 P: 1×1 F: 3×3	Output: 8×16 maps: 128 S: 1×1 P: 1×1 F: 3×3	Output: 16×16 maps: 128 S: 1×1 P: 1×1 F: 3×3
Max-pooling 2	Output: 4×8 maps: 128 S: 1×2 K: 1×2	Output: 8×8 maps: 128 S: 1×2 K: 1×2	Output: 8×8 maps: 128 S: 2×2 K: 2×2
Convolution 3	Output: 4×8 maps: 256 S: 1×1 P: 1×1 F: 3×3	Output: 8×8 maps: 256 S: 1×1 P: 1×1 F: 3×3	Output: 8×8 maps: 256 S: 1×1 P: 1×1 F: 3×3
Max-pooling 3	Output: 4×4 maps: 256 S: 1×2 K: 1×2	Output: 4×4 maps: 256 S: 2×2 K: 2×2	Output: 4×4 maps: 256 S: 2×2 K: 2×2
Convolution 4	Output: 4×4 maps: 512 S: 1×1 P: 1×1 F: 3×3	Output: 4×4 maps: 512 S: 1×1 P: 1×1 F: 3×3	Output: 4×4 maps: 512 S: 1×1 P: 1×1 F: 3×3
Max-pooling 4	Output: 2×2 maps: 512 S: 2×2 K: 2×2	Output: 2×2 maps: 512 S: 2×2 K: 2×2	Output: 2×2 maps: 512 S: 2×2 K: 2×2
Convolution 5	Output: 2x2 maps: 512 S: 1×1 P: 1×1 F: 3×3	Output: 2x2 maps: 512 S: 1×1 P: 1×1 F: 3×3	Output: 2x2 maps: 512 S: 1×1 P: 1×1 F: 3×3
Max-pooling 5	Output: 1×1 maps: 512 S: 2×2 K: 2×2	Output: 1×1 maps: 512 S: 2×2 K: 2×2	Output: 1×1 maps: 512 S: 2×2 K: 2×2

1×2 or 2×2 pixels without padding. Finally, the CNN model generates $1 \times 1 \times 512$ feature maps, which will be concatenated into a single vector. This later contains the features extracted from the input frame. To avoid the training problems of the deep convolutional layers, the batch normalization [61] technique is used. Two batch normalization layers are added after the 3rd and 4th convolutional layers respectively. This technique speeds up the training process and reduces overfitting. This operation will be repeated for the remaining frames. At the end of this step, a sequence of feature vectors will represent each text line image, which is the input of the recurrent neural network employed in the classification step.

2.5 Classification

The classification and recognition phase is often the last step, which involves assigning a label, or class, to the features vector. Generally, in recognition systems the classification process requires a preliminary training stage, also called learning, where the classifier learns from the training samples, allowing it in the classification phase to predict classes of new samples. In this work, the sequence of feature vectors obtained in the previous step is used in the classification process. The latter makes it possible to assign the appropriate transcription to the sequence of vectors. In this paper, we propose an architecture that consists of a bidirectional recurrent neural network (BRNN) with the gated recurrent units (GRU) block [11] and the Connectionist temporal classification (CTC) layer [18]. This architecture will allow to build a model that can learn how to classify each sequence of vectors.

The Recurrent Neural Network (RNN) is a deep learning architecture built from artificial neural networks originally proposed by J.L. Elman [13]. RNN differs from the classical neural network because it includes cyclic connections, which can model contextual information of a sequence dynamically. The output of an RNN is calculated using both the input at the current instant t and the output of the hidden layer at the instant $t-1$. Specifically, consider an RNN with X input neurons, H hidden neurons, and Y output neurons. At a time t, the state of the hidden layer $h_{(t-1)}$ at the time $t-1$ and the input x_t at time t are passed to an activation function in order to calculate the state of the hidden layer h_t at time t. The following equation formalizes this function:

$$h_t = f(w_{hh}h_{t-1} + w_{xh}x_t + b) \tag{1}$$

Where $f(.)$is a nonlinear activation function (Sigmoid, Tanh, Relu, ...), w_{hh} is the weight matrix that links the hidden layer at time $t-1$ and the hidden layer at time t, and w_{xh} is the weight matrix that links the input layer with the hidden layer at time t, and b is the bias vector of the hidden layer at time t. Then, the output y_t at time t is calculated as follows:

$$y_t = f(w_{hy}h_t + b) \tag{2}$$

Where w_{hy} correspond to the hidden-to-output weights matrix. One of the ideas introduced in the literature to improve the amount of context information is to exploit at time t the time dependencies of both the past and the future. Indeed, this is done using an RNN model with two hidden layers; one traverses the input sequence from left to right, while the other runs it in the opposite direction. This model is known as bidirectional recurrent neural network BRNN [48]. The states of the hidden layer and the output layer in this model are calculated as follows:

$$h_t^b = f(w_{h^b h^b}h_{t-1}^b + w_{xh^b}x_t + b) \tag{3}$$

$$h_t^f = f(w_{h^f h^f}h_{t+1}^f + w_{xh^f}x_t + b) \tag{4}$$

$$y_t = f(w_{h^b h}h_{t-1}^b + w_{xh^f}x_t^f + b) \tag{5}$$

Where h_t^p and h_t^f respectively represent the hidden layers in the backwards forwards directions.

The BRNN is trained in the same way as a classical neural network, with the use of the backpropagation algorithm. However, whatever the learning algorithm used, the main disadvantage of a BRNN is the disappearance of the gradient [19] (Vanishing Problem). Indeed, we see a very rapid decrease of the gradient during the retro-propagation, which makes learning long-term dependencies very difficult. One solution proposed to avoid this problem is to add the memory block mechanism to the BRNN. This block is positioned at the hidden layer, and includes one or more memory units that give the network the ability to memorize or forget about long-term or short-term information. Two types of block are found in the literature: the first is LSTM (Long Short-Term Memory) [21] and the second is GRU (Gated Recurrent Unit) [11].

The LSTM block consists of a memory cell for storing information and three control gates: an input gate, an output gate and a forget gate (see Fig. 14a). The input gate controls the importance of the state of the current input. A state is considered relevant if this gate gives a value close to 1. The importance of the previous state on the current state of the memory cell is controlled by the forget gate. The output gate controls the importance of the current state on the rest of the network (higher layers and next time steps). Generally, LSTM introduces a linear dependence between the memory cell c_t and its previous $c_{(t-1)}$

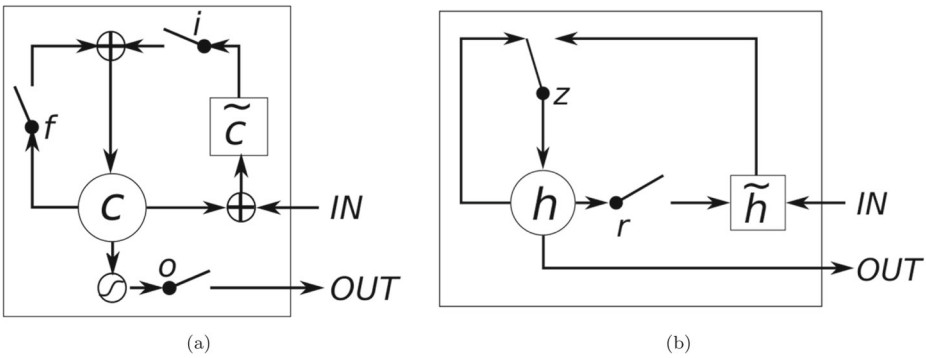


Fig. 14 Visualization of Long short-term memory (LSTM) and Gated Recurrent Unit (GRU) architecture [11]. **a**: LSTM gates: input (**I**), forget (**f**) and output (**o**). **b**: GRU gates: update (**z**) and reset (**r**)

at each hidden layer in order to control the information flow in the network. The operations of a RNN with LSTM block are written as follows:

$$i_t = \sigma(w_{hi}h_{t-1} + w_{xi}x_t + b_i) \tag{6}$$
$$f_t = \sigma(w_{hf}h_{t-1} + w_{xf}x_t + b_f) \tag{7}$$
$$o_t = \sigma(w_{ho}h_{t-1} + w_{xo}x_t + b_o) \tag{8}$$
$$g_t = \tanh(w_{xg}x_t + w_{hg}h_{t-1} + b_g) \tag{9}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(w_{hh}h_{t-1} + w_{xh}x_t + b_c) \tag{10}$$
$$h_t = o_t \odot \tanh(c_t) \tag{11}$$

Where i_t, f_t , o_t and h_t respectively denote the values of input gate, output gate, forget gate and output hidden layer state at time t. The symbol $\odot$ represents the element-wise product of vectors. w denote the weight matrices connecting different gates and layers, and b denote the corresponding bias vectors.

The GRU [11] block has recently been proposed as a simpler alternative to the LSTM unit, while sharing the same objective of avoiding long-term dependency problems. It contains two gates that make it possible to estimate the flow of information inside the block, but, it does not contain a separate memory cell. The gates of a GRU are: the Update Gate and Reset Gate (see Fig. 14b). The update gate role is similar to the role of the LSTM's forget gate, it controls the importance of the previous state on the current state of the network, while, the reset gate allows the GRU block to ignore information that is not relevant in the next time steps. The GRU block is usually defined by the following equations:

$$z_t = \sigma(w_{hz}h_{t-1} + w_{xz}x_t + b_z) \tag{12}$$
$$r_t = \sigma(w_{hr}h_{t-1} + w_{xr}x_t + b_r) \tag{13}$$
$$\tilde{h}_t = \tanh(w_{xh}x_t + w_{hh}(r_t \odot h_{t-1}) + b_h) \tag{14}$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{15}$$

Where z_t, r_t, $\tilde{h}_t$ and h_t are respectively the update gate, the update gate, the current block state and the output hidden layer state.

A typically RNN contains a very large number of parameters; the bad adjustment of these parameters can drive to overfitting. This means that the network work effectively on the

training data, but does not generalize well on the testing data. To remedy this problem, the dropout layer has been introduced [52]. This layer is used during learning. At each iteration, a certain percentage of neurons of a layer are randomly disabled to artificially reduce the number of parameters. This way allows the RNN to learn more generalized parameters that do not focus on the details of the learning dataset. Once the learning is complete, all the neurons are reactivated.

Even if an RNN with a GRU or LSTM memory block is able to model long-term dependencies, it requires a pre-segmented training data to allow the network to learn to provide the correct output at every time step. Therefore, the input feature vector x_t at each instant t has to be assigned to the corresponding output target (corresponding character class). However, in the case of text recognition, no character segmentation is performed, and the feature sequences that are extracted with a sliding window are not separated. Gravex et al. [18] provided a solution to this problem by introducing a specific layer, termed connectionist temporal classification (CTC) layer, which extends the use of an RNN for the case of an unsegmented data sequence. The essential role of the CTC layer is to calculate at each time-step the posterior probability of an output (character class) for each unsegmented input sequence. It only requires presenting the feature vectors sequence side by side the target sequence of characters during the training phase.

Figure 15 shows the general architecture used in the classification step. It takes as input the text line image and gives as an output the corresponding text transcription without having to perform an explicit segmentation

3 Experimental results

This section is devoted to presenting the results. We start by giving an overview of the database and the evaluation measures used. Next, we study the impact of different parameters and configurations on the results of the recognition. Finally, we compare the results obtained with those obtained by other existing systems on the same database.

3.1 Datasets and evaluation measures

In order to evaluate the performance of the proposed system, we perform the experiments on the ICDAR2015 Smartphone document OCR dataset [7]. This database contains 12100 images, which are separated into two sub-bases: the sample database (3630 images) and the test database (8470). The text format is grouped as a single column with a font and a particular size in each image of the database. The images are acquired by Two smartphones (Samsung Galaxy $S4$ and Nokia Lumia 920) in diverse conditions of lighting, blur and perspective. In our study, the training phase is carried out on the Sample dataset, which have a total of 162186 lines after segmentation, while the testing phase is performed using test set (335740 lines).

All our experiments were performed on an Ubuntu 16.04 LTS (64 bit) operating system installed on a computer with Intel Core $i7-4770$ CPU, 16 Go RAM and NVIDIA GeForce GTX 980 $4GO$. Matlab $r2014$ environment is used to implement the steps of preprocessing, baseline correction and normalization. In respect of the features extraction and classification step, we have used the powerful deep learning framework $Torch7$.

The recognition accuracy percentage (CRA %) metric is used to evaluate the performance of the proposed system. The CRA is determined using the editing distance (Levenshtein Distance). This distance is estimated by counting the minimum number of characters that

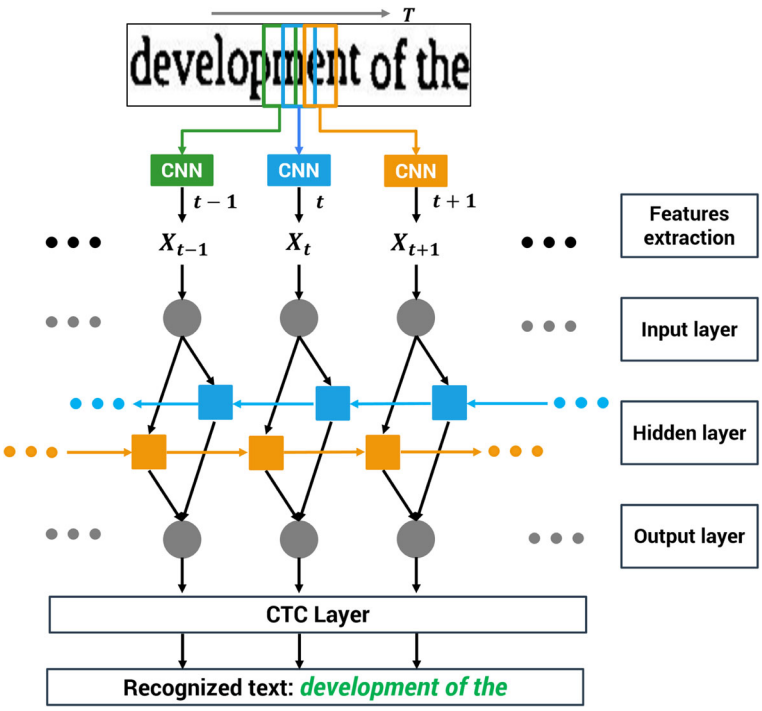


Fig. 15 Figure 7: General overview of the CNN-BRNN architecture. The architecture includes three main parts: 1) CNN model that extracts a sequence of feature vectors from the input text-line image; 2) BRNN network (input, hidden and output layers), which predicts the corresponding class of each feature vector at each time step; 3) CTC layer, which transforms the sequence of output predictions into the final sequence of characters

can be inserted, deleted, and substituted to correct the result text of the recognition phase and have the same text of the corresponding terrain truth. Therefore, the CRA is calculated as follows:

$$CRA = \frac{n - ED}{n} \times 100 \tag{16}$$

$$Edit\ distance = ED = I + D + S \tag{17}$$

Where n represent the total number of characters in the ground truth text, I, D and S are respectively the numbers of characters inserted, deleted and substituted in the output sequence of text of the proposed system.

3.2 Parameter Setup

The optimal performance of a system implies a rigorous selection of parameters. In our case, four parameters are to be adjusted to obtain optimal values. The first parameter refers to the CNN model used in the features extraction step. The following parameter is related to the architecture of RNN (unidirectional or bidirectional) used in the classification step. The other two are related to the type of the memory block (GRU or LSTM) and the number of memory units in the hidden layer of RNN.

Table 2 Comparison of character recognition accuracy with different CNN models and RNN architectures

Features extraction	Classification	Training (%)	Test (%)
$CNN_{4\times32}$	RNN	86.40	78.16
	BRNN	90.73	84.96
$CNN_{8\times32}$	RNN	92.32	88.03
	BRNN	95.80	94.49
$CNN_{16\times32}$	RNN	90.59	87.21
	BRNN	95.18	92.13

The first evaluation focuses on the two first parameters. Indeed, we have evaluated and compared three CNN models: $CNN_{4\times32}$, $CNN_{8\times32}$ and $CNN_{16\times32}$. The three models are used in the features extraction phase, and each one of the three was combined in the classification stage with either a unidirectional recurrent neural network RNN or a bidirectional recurrent neural network BRNN. Therefore, we will have 6 architectures that will be evaluated on the training and testing datasets. Results obtained by different architectures are presented in Table 2. On the basis of these results, we can notice that the model $CNN_{8\times32}$ has achieved the best performance compared to the models $CNN_{4\times32}$ and $CNN_{16\times32}$. More precisely, the first model $CNN_{4\times32}$ combined with RNN or BRNN has achieved a CRA of 78.16% and 84.96% respectively. The second model $CNN_{8\times32}$ combined with RNN or BRNN has reached a CRA of 88.03% and 94.49% respectively. While the third model ($CNN_{16\times32}$) that has combined with RNN or BRNN has achieved a CRA of 87.21% and 92.13% respectively. These results clearly demonstrate that applying a deep convolutional neural network on a frame with a size of 4×32 pixels not perform better. This is due to the fact that this frame is small in size and poor in content, which will generate features that wouldn't aid to discriminate correctly the classes of characters in the classification step. We can also note that increasing the size of a frame will not necessarily improve significantly the CRA, and can demand a large amount of computation. Regarding the classification step, the results prove that unidirectional RNN perform worse than bidirectional BRNN. This confirms that using the past context side by side with the future context is important to predict the appropriate transcription of the input sequence of feature vectors. Therefore, we decide to combine the $CNN_{8\times32}$ model with $BRNN$ architecture to carry out the next experiments.

The main purpose of the second experiment is to study the influence of choice of the memory block (GRU or LSTM) and the number of memory units in the hidden layer (size of the hidden layer) on both the character recognition accuracy and the time consuming during the training phase. For this reason, we conducted a series of four tests with two models of a BRNN; one model contains the GRU memory block, and the other LSTM. For each model, we test various hidden layer sizes: 100, 150, 200 and 250. These sizes are chosen on the one hand to minimize the recognition error rate of a $BRNN-GRU$ or $BRNN-LSTM$, and on the other hand to find the optimal size.

Figure 16 illustrates the comparison of the recognition error rate curves of the $BRNN-GRU$ and $BRNN-LSTM$ models with different sizes. The training time after 20000 iterations with respect to the size of the hidden layer is shown in Fig. 17.

From the results, it can be noted that the type and number of the memory blocks in the hidden layer affect the recognition accuracy in the different tests, and it improves with the increase of the size of the hidden layer. Indeed, we can deduce from Figs. 16 and 17 two

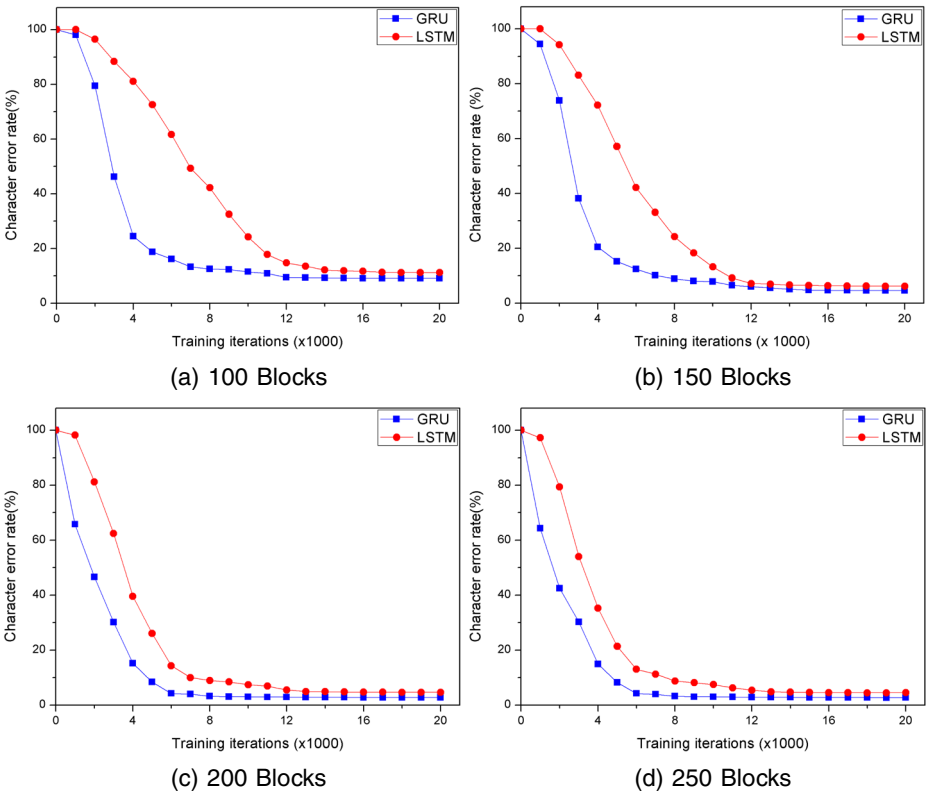


Fig. 16 Recognition error curves as a function of the type of memory block and the size of hidden layer

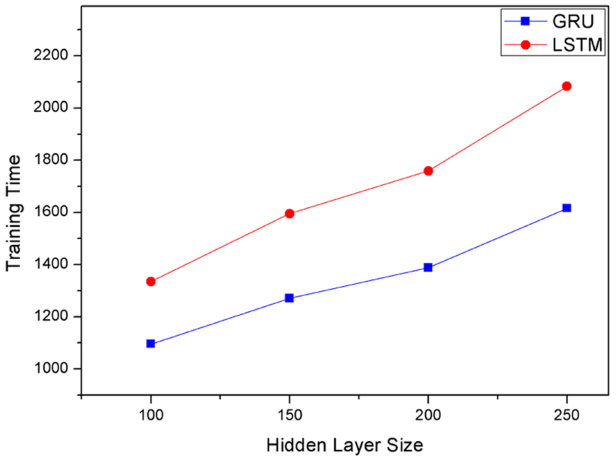


Fig. 17 Training time as a function of the type of memory block and the size of hidden layer

Table 3 Character recognition accuracy on training and test dataset using CNN8x32 BRNN-LSTM or CNN8x32 BRNN-GRU architecture with different hidden layer sizes

Architecture	Hidden layer size	Training (%)	Test (%)
CNN_{4x32} BRNN-LSTM	100	91.98	88.79
	150	94.66	93.84
	200	96.54	95.36
	250	96.71	95.52
CNN_{4x32} BRNN-GRU	100	94.50	90.92
	150	98.40	95.44
	200	98.69	97.26
	250	98.90	97.28

remarks; first, the $BRNN - GRU$ model outperforms the $BRNN - LSTM$ model and achieves better results in recognition rate. Second, the GRU block converges faster than the LSTM block and makes it possible to obtain a high recognition rate with less iteration. In addition, we note that increasing the size of the hidden layer decreases the error rate, but at the same time, this will lead to increasing the time required for the training phase.

Table 3 shows the results of all experiments on the Sample dataset after 20000 iterations. The best result is achieved by the model $BRNN - GRU$ that contains a hidden layer of size 250 units with a CRA of 97.28%, without any significant improvement with more units. Therefore, we decided to choose the memory block GRU and 200 as the optimal size of the hidden layer for the rest of this work.

3.3 Comparison with Existing Systems

In this experiment, we evaluate the capacity and robustness of the proposed system by comparing it to the results of the groups that participated in the ICDAR2015 competition of smartphone documents OCR [7]. We conducted this comparison by using the same $CNN_{8\times32} - BRNN - GRU$ system that was evaluated in the previous experiments. The character accuracy obtained by the proposed system as well as the participated systems in the ICDAR2015 challenge, are presented Table 4.

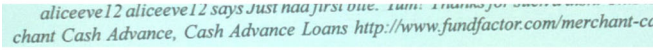
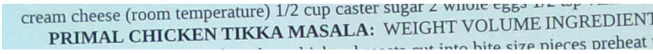
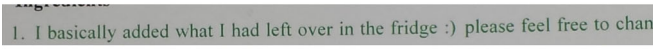
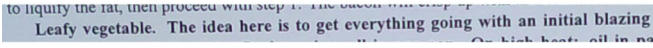
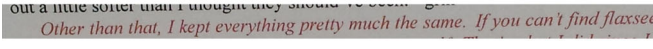
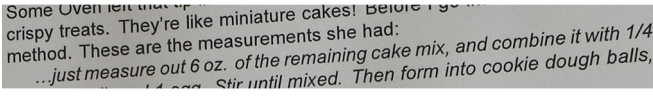
The CCC system uses a $RNN - LSTM$ model that was trained on both sharp and blurry text-line image. A $RNN - LSTM$ model that trained only on binary text line image has been employed in the A2iA method. The methods proposed by LDRE and Digiform use the Abbyy Finereader Engine, whereas CartPerk use the Tesseract OCR Engine to perform the

Table 4 Comparison of the proposed architecture and existing methods in terms of character accuracy

System	Character accuracy (%)	# Errors per Page
CCC	99.93	2
LRDE	95.85	120
Digiform	95.33	135
A2iA	93.84	178
CartPerk	91.19	254
Finereader	87.61	357
Proposed system	97.28	78

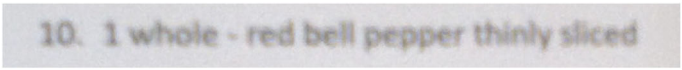
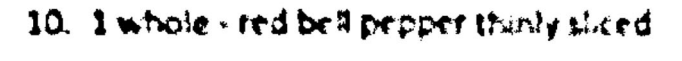
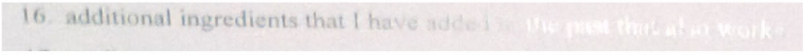
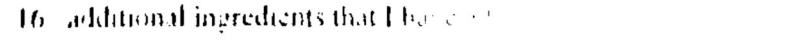
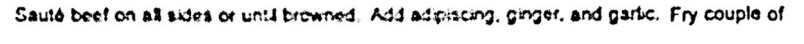
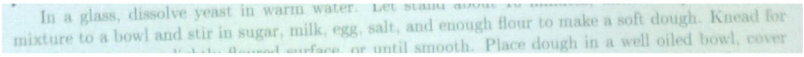
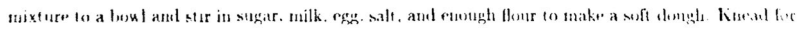
recognition. All the above-mentioned methods perform the recognition task after applying the preprocessing stage, while the Finereader method carry out the recognition with Abbyy Finereader Engine 11 without applying any pre- or post-processing method. All method use non-learned features that are extracted using hand-crafted feature-based methods.

Table 5 Successful examples. For each example, we show the input text-line image from the original document image (input), the result after the preprocessing stage (Pre-p.), the output after the classification stage (output) and the ground truth (G.T.)

Input	
Pre-p.	**1 How to make congue Russian / Polish Salad**
Output	1 How to make congue Russian / Polish Salad
G.T.	1 How to make congue Russian / Polish Salad
Input	
Pre-p.	***chant Cash Advance, Cash Advance Loans http://www.fundfactor.com/merchant-cash-advance.html***
Output	Cash Advance, Cash Advance Loans http:/www.fundfactor.com/merchant-cash-advance.html
G.T.	Cash Advance, Cash Advance Loans http://www.fundfactor.com/merchant-cash-advance.html
Input	
Pre-p.	**PRIMAL CHICKEN TIKKA MASALA: WEIGHT VOLUME INGREDIENT SCALING IN-**
Output	PRIMAL CHICKEN TIKKA MASALA; WEIGHT VOLUME INGREDIENT SCALING IN-
G.T.	PRIMAL CHICKEN TIKKA MASALA: WEIGHT VOLUME INGREDIENT SCALING IN-
Input	
Pre-p.	**1. I basically added what I had left over in the fridge :) please feel free to change it up and add**
Output	1. I basically added what I had left over in the fridge :I please feel free to change it up and add
G.T.	1. I basically added what I had left over in the fridge :) please feel free to change it up and add
Input	
Pre-p.	**Leafy vegetable. The idea here is to get everything going with an initial blazing sautee step, then**
Output	Leafy vegetable. The idea here is to get everything going with an initial blazing sautee step. then
G.T.	Leafy vegetable. The idea here is to get everything going with an initial blazing sautee step, then
Input	
Pre-p.	***Other than that, I kept everything pretty much the same. If you can't find flaxseed meal, you can***
Output	Other than that, I kept everything pretty much the same. If you can't find flaxseed meal, you can
G.T.	Other than that, I kept everything pretty much the same. If you can't find flaxseed meal, you can
Input	
Pre-p.	***…just measure out 6 oz. of the remaining cake mix, and combine it with 1/4 cup vegetable***
Output	.. just measure out 6 oz. of the remaining cake mix, and combine it with 1/4 cup vegetable
G.T.	...just measure out 6 oz. of the remaining cake mix, and combine it with 1/4 cup vegetable

We notice from the results of all participated methods that the CCC method provides the best character rate with 99.93% compared to our method which reached 96, 76%. The high rate of the CCC method can be explained by the fact that the $RNN - LSTM$ employed is trained use a data augmentation technique. Which mean that the classifier is trained using both the images before and after the preprocessing step. However, as we mentioned earlier the a high computational cost will be required to train the LSTM in addition, it has trained more times with two types of images and thus would make it computationally very demanding. The proposed method is more precise compared to the other methods that use an OCR engine tools (LRDE, Digiform, CartPerk, Finereader) and also the A2iA method

Table 6 Failure examples. For each example, we show the input text-line image from the original document image (input), the result after the preprocessing stage (Pre-p.), the output after the classification stage (output) and the ground truth (G.T.)

Input	
Pre-p.	
Output	10 1 cole , nd bel pepgr thmly ucd
G.T.	10. 1 whole - red bell pepper thinly sliced
Input	
Pre-p.	
Output	16 adltonal inaredicnta thh l l
G.T.	16. additional ingredients that I have added in the past that also works
Input	
Pre-p.	
Output	abitur varlus fringilla nisl. Dms prct iu n auaue. Nem wlputate
G.T.	abitur varius fringilla nisl. Duis pretium mi euismod erat. Maecenas id augue. Nam vulputate.
Input	
Pre-p.	
Output	Sauta beel on al ubee or unll Acwrad. Atd adpscwq, giogor, ond gartc. Fry couplo of
G.T.	Saut beef on all sides or until browned. Add adipiscing, ginger, and garlic. Fry couple of
Input	
Pre-p.	
Output	mixtnre lo a bovl and stn in sugar, mill eg sal and enwgh flour o mae suft dwqli. Annl lw
G.T.	mixture to a bowl and stir in sugar, milk, egg, salt, and enough flour to make a soft dough. Knead for

that perform the recognition with a $RNN-LSTM$ model. In addition, we show the advantage of using the learned features that are extracted by a CNN model. CNN extracts more efficiently and automatically provides the most relevant features thanks to its deep architecture. It achieves better performance than other methods which incorporate hand-crafted and non-learned features.

Tables 5 and 6 illustrate some examples of preprocessing and recognition results when the proposed system is applied to various input text-line images. In Table 5, the text contained in the image-lines has been successfully preprocessed and recognized in different cases of sizes, colors, styles, lights, perspective distortions and presence of skewed or slanted degradation. In Table 6, the input images are affected by focus or motion blur because of unfocused camera. Consequently, the recognition errors are occurred due to degraded text quality.

4 Conclusion

In this paper, we have presented a novel system based on a combination of Deep Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) in order to build a text recognition architecture in the document images obtained by a smartphone. The system begins with a preprocessing step that detects and prepares the text-lines images. Then, different CNN models were explored to extract discriminative features from each text-line image. Moreover, we utilize a BRNN that integrated a GRU or LSTM memory block and compare the character recognition accuracy results with the various size of hidden layer. The experiments indicate that the $CNN_{8\times32}$ performs better when combined with the $BRNN-GRU$ architecture and achieves a high recognition rate with less iteration and computation time.

In the future work, we would like to improve the recognition results by adding more efficient methods to address the problem of blur distortion. We are also planning to propose a language model and a dictionary, which will be integrated into the post-processing phase and will lead to increased recognition rates as well.

References

1. Ahmad I, Rothacker L, Fink GA, Mahmoud SA (2013) Novel sub-character hmm models for arabic text recognition. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR), IEEE, pp 658–662
2. Antonacopoulos A, Clausner C, Papadopoulos C, Pletschacher S (2015) Icdar2015 competition on recognition of documents with complex layouts-rdcl2015. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), IEEE, pp 1151–1155
3. Bahi HE, Zatni A (2017) Segmentation and recognition of text images acquired by a mobile phone. International Journal of Tomography & Simulation^{TM} 30(4):95–107
4. Banumathi KL, Jagadeesh Chandra AP (2016) Line and word segmentation of kannada handwritten text documents using projection profile technique. In: 016 international conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), IEEE, pp 196–201
5. Bertolami R, Bunke H (2008) Hidden markov model-based ensemble methods for offline handwritten text line recognition. Pattern Recogn 41(11):3452–3460
6. Bukhari SS, Shafait F, Breuel TM (2011) Improved document image segmentation algorithm using multiresolution morphology. In: Document Recognition and Retrieval XVIII, vol 7874. International Society for Optics and Photonics, pp 78740D

7. Burie J-C, Chazalon J, Coustaty M, Eskenazi S, Luqman MM, Mehri M, Nayef N, Ogier J-M, Prum S, Rusiñol M (2015) Icdar2015 competition on smartphone document capture and ocr (smartdoc). In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), IEEE, pp 1161–1165
8. Canny J (1987) A computational approach to edge detection. In: Readings in Computer Vision, Elsevier, pp 184–203
9. Castro DMR, Revel A, Ménard M (2015) Document image analysis by a mobile robot for autonomous indoor navigation. in: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), IEEE, pp 156–160
10. Chen S, Zhang C, Dong M (2018) Deep age estimation: From classification to ranking. IEEE Transactions on Multimedia, 20(8)
11. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555
12. El Bahi H, Zatni A (2016) Pre-processing of document images obtained with a smartphone. International Review on Computers and Software 11(12):1187–1198
13. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
14. Eskenazi S, Gomez-Krämer P, Ogier J-M (2017) A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recogn 64:1–14
15. Espana-Boquera S, Castro-Bleda MJ, Gorbe-Moya J, Zamora-Martinez F (2011) Improving offline handwritten text recognition with hybrid hmm/ann models. IEEE Trans Pattern Anal Mach Intell 33(4):767–779
16. Gllavata J, Ewerth R, Freisleben B (2004) Text detection in images based on unsupervised classification of high-frequency wavelet coefficients. In: ICPR 2004. Proceedings of the 17th International Conference on Pattern Recognition, 2004, vol 1. IEEE, pp 425–428
17. Granell E, Chammas E, Likforman-Sulem L, Martínez-Hinarejos CD, Mokbel C, Cîrstea B-I (2018) Transcription of spanish historical handwritten documents with deep neural networks. Journal of Imaging 4(1):15
18. Graves A, Fernández S, Gomez F, Schmidhuber J (2006) Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, ACM, pp 369–376
19. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Netw 18(5-6):602–610
20. Graves A, Schmidhuber J (2009) Offline handwriting recognition with multidimensional recurrent neural networks. in: Advances in Neural Information Processing Systems, pp 545–552
21. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
22. Huang W, Yu Q, Tang X (2014) Robust scene text detection with convolution neural network induced mser trees. in: European Conference on Computer Vision, Springer, pp 497–511
23. Keysers D, Deselaers T, Gollan C, Ney H (2007) Deformation models for image recognition. IEEE Trans Pattern Anal Mach Intell 29(8):1422–1435
24. Keysers D, Deselaers T, Rowley HA, Wang L-L, Carbune V (2017) Multi-language online handwriting recognition. IEEE Trans Pattern Anal Mach Intell 39(6):1180–1194
25. Khare V, Shivakumara P, Raveendran P (2015) A new histogram oriented moments descriptor for multi-oriented moving text detection in video. Expert Syst Appl 42(21):7627–7640
26. Kim BS, Koo HI, Cho NI (2015) Document dewarping via text-line based optimization. Pattern Recogn 48(11):3600–3614
27. Kozielski M, Doetsch P, Ney H (2013) Improvements in rwth's system for off-line handwriting recognition. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR), IEEE, pp 935–939
28. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
29. Li J, Liang X, Shen SM, Xu T, Feng J, Yan S (2018) Scale-aware fast r-cnn for pedestrian detection. IEEE Trans Multimedia 20(4):985–996
30. Liang J, Doermann D, Li H (2005) Camera-based analysis of text and documents: a survey. Int J Doc Anal Recognit (IJDAR) 7(2-3):84–104
31. Liu C, Yu Z, Wang B, Ding X (2015) Restoring camera-captured distorted document images. Int J Doc Anal Recognit (IJDAR) 18(2):111–124
32. Liu X, Wang W (2015) An effective graph-cut scene text localization with embedded text segmentation. Multimed Tools Appl 74(13):4891–4906
33. Liu Z, Zhang C, Tian Y (2016) 3d-based deep convolutional neural network for action recognition with depth sequences. Image Vis Comput 55:93–100
34. Maalej R, Tagougui N, Kherallah M (2016) Online arabic handwriting recognition with dropout applied in deep recurrent neural networks. In: 2016 12th IAPR Workshop on Document Analysis Systems (DAS), IEEE, pp 417–421

35. Messina R, Louradour J (2015) Segmentation-free handwritten chinese text recognition with lstm-rnn. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), IEEE, pp 171–175
36. Morillot O, Likforman-Sulem L, Grosicki E (2013) New baseline correction algorithm for text-line recognition with bidirectional recurrent neural networks. J Electron Imaging 22(2):023028
37. Nagabhushan P, Alaei A (2010) Tracing and straightening the baseline in handwritten persian/arabic text-line: a new approach based on painting-technique. Int J Comput Sci Eng 2(4):907–916
38. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp 807–814
39. Namboodiri AM, Jain AK (2007) Document structure and layout analysis. In: Digital Document Processing, Springer, pp 29–48
40. Nayef N, Luqman MM, Prum S, Eskenazi S, Chazalon J, Ogier J-M (2015) Smartdoc-qa: a dataset for quality assessment of smartphone captured document images-single and multiple distortions. in: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), IEEE, pp 1231–1235
41. Naz S, Umar AI, Ahmad R, Ahmed SB, Shirazi SH, Siddiqi I, Razzak MI (2016) Offline cursive urdu-nastaliq script recognition using multidimensional recurrent neural networks. Neurocomputing 177:228–241
42. Razak Z, Zulkiflee K, Idris MYI, Tamil EM, Noor MNM, Salleh R, Yaakob M, Yusof ZM, Yaacob M (2008) Off-line handwriting text line segmentation: a review. International Journal of Computer Science and Network Security 8(7):12–20
43. Rehman A, Saba T (2014) Neural networks for document image preprocessing: state of the art. Artif Intell Rev 42(2):253–273
44. Retsinas G, Louloudis G, Stamatopoulos N, Gatos B (2016) Keyword spotting in handwritten documents using projections of oriented gradients. In: 2016 12th IAPR Workshop on Document Analysis Systems (DAS), IEEE, pp 411–416
45. Roy PP, Bhunia AK, Das A, Dey P, Pal U (2016) Hmm-based indic handwritten word recognition using zone segmentation. Pattern Recogn 60:1057–1075
46. Saha S, Basu S, Nasipuri M (2015) ilpr: an indian license plate recognition system. Multimed Tools Appl 74(23):10621–10656
47. Scherer D, Müller A, Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. In: Artificial Neural Networks–ICANN 2010, Springer, pp 92–101
48. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
49. Shekar BH, Smitha ML, Shivakumara P (2014) Discrete wavelet transform and gradient difference based approach for text localization in videos. In: 2014 fifth International Conference on Signal and Image Processing (ICSIP), IEEE, pp 280–284
50. Shi B, Bai X, Yao C (2017) 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 39(11):2298–2304
51. Smith RW (2009) Hybrid page layout analysis via tab-stop detection. In: 2009 10Th International Conference on Document Analysis and Recognition, IEEE, pp 241–245
52. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
53. Su B, Lu S (2014) Accurate scene text recognition based on recurrent neural network. In: Asian Conference on Computer Vision, Springer, pp 35–48
54. Sueiras J, Ruiz V, Sanchez A, Velez JF (2018) Offline continuous handwriting recognition using sequence to sequence neural networks. Neurocomputing 289:119–128
55. Tang Y, Wu X, Bu W (2014) Text line segmentation based on matched filtering and top-down grouping for handwritten documents. In: 2014 11th IAPR International Workshop on Document Analysis Systems (DAS), IEEE, pp 365–369
56. Tran TA, Na IS, Kim SH (2016) Page segmentation using minimum homogeneity algorithm and adaptive mathematical morphology. Int J Doc Anal Recognit (IJDAR) 19(3):191–209
57. Ullah A, Ahmad J, Muhammad K, Sajjad M, Baik SW (2018) Action recognition in video sequences using deep bi-directional lstm with cnn features. IEEE Access 6:1155–1166
58. Wang X, Song Y, Zhang Y, Xin J (2017) A hierarchical recursive method for text detection in natural scene images. Multimed Tools Appl 76(24):26201–26223
59. Wang X, Yi G, Wang Y, Yu J (2017) Automatic breast tumor detection in abvs images based on convolutional neural network and superpixel patterns. Neural Comput Applic, pp 1–13
60. Wei Y, Xia W, Lin M, Huang J, Ni B, Dong J, Zhao Y, Yan S (2016) Hcp: a flexible cnn framework for multi-label image classification. IEEE Trans Pattern Anal Mach Intell 38(9):1901–1907

61. Yan C, Xie H, Chen J, Zha Z, Hao X, Zhang Y, Dai Q (2018) Cross-modality bridging and knowledge transferring for image understanding. IEEE Trans Multimedia Early Access
62. Yan C, Xie H, Chen J, Zha Z, Hao X, Zhang Y, Dai Q (2018) A fast uyghur text detector for complex background images. IEEE Trans Multimedia 20(12):3389–3398
63. Yan C, Xie H, Liu S, Yin J, Zhang Y, Dai Q (2018) Effective uyghur language text detection in complex background images for traffic prompt identification. IEEE Trans Intell Transp Syst 19(1):220–229
64. Ye Q, Doermann D (2015) Text detection and recognition in imagery: a survey. IEEE Trans Pattern Anal Mach Intell 37(7):1480–1500
65. Yin X-C, Zuo Z-Y, Tian S, Liu C-L (2016) Text detection, tracking and recognition in video: a comprehensive survey. IEEE Trans Image Process 25(6):2752–2773
66. Yousfi S, Berrani S-A, Garcia C (2017) Contribution of recurrent connectionist language models in improving lstm-based arabic text recognition in videos. Pattern Recogn 64:245–254
67. Zhang Y-D, Dong Z, Chen X, Jia W, Du S, Muhammad K, Wang S-H (2017) Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. Multimedia Tools and Applications, pp 1–20
68. Zhu Y, Yao C, Bai X (2016) Scene text detection and recognition: recent advances and future trends. Front Comp Sci 10(1):19–36
69. Zhu Y, Zhang K (2017) Text segmentation using superpixel clustering. IET Image Process 11(7):455–464

Hassan El Bahi received a Ph.D. degree in the field of Artificial Intelligence and Image Processing, as well as a Master's in Software Engineering of Distributed Systems from the Faculty of sciences University Ibn Zohr, Agadir, Morocco. He is currently an assistant professor in the national school of business and management at Cadi Ayyad University, Marrakech, Morocco. His research interests involve machine learning, deep learning, document image processing and recommender systems.

Abdelkarim Zatni was educated at the Telecom Bretagne University France; He obtained a PhD at the National School of Engineers of Brest France in 1994. He has been teaching for 28 years. He is currently a Professor in the department of physics of the faculty of sciences, Ibn Zohr University. He conducts his research and teaches in computer science and Telecommunications.