


Neural Networks Pipeline for Offline Machine Printed Arabic OCR

Mohamed A. Radwan¹ · Mahmoud I. Khalil¹  ·
Hazem M. Abbas¹

© Springer Science+Business Media, LLC 2017

Abstract In the context of Arabic optical characters recognition, Arabic poses more challenges because of its cursive nature. We purpose a system for recognizing a document containing Arabic text, using a pipeline of three neural networks. The first network model predicts the font size of an Arabic word, then the word is normalized to an 18pt font size that will be used to train the next two models. The second model is used to segment a word into characters. The problem of words segmentation in the Arabic language, as in many similar cursive languages, presents a challenge to the OCR systems. This paper presents a multichannel neural network to solve the offline segmentation of machine-printed Arabic documents. The segmented characters are then fed as an input to a convolutional neural network for Arabic characters recognition. The font size prediction model produced a test accuracy of 99.1%. The accuracy of the segmentation model using one font is 98.9%, while four-font model showed 95.5% accuracy. The whole pipeline showed an accuracy of 94.38% on Arabic Transparent font of size 18pt from APTI data set.

Keywords OCR · Arabic word segmentation · Convolutional neural networks · Character recognition

1 Introduction

An optical character recognition system is used to analyze a printed document, and convert it into a computer readable text format. This format can be accessible by information retrieval systems, thus easing the access to a vast amount of information. Arabic documents are more challenging than other Latin languages since they have very different and complex characteristics. Automatic segmentation and recognition of Arabic documents have always been a tough problem to solve [1]. Unlike Latin languages which are cursive mostly in

✉ Mahmoud I. Khalil
mahmoud.khalil@eng.asu.edu.eg

¹ Computers and Systems Engineering Department, Faculty of Engineering, Ain Shams University, Cairo, Egypt

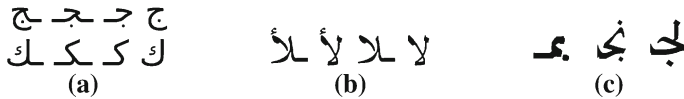


Fig. 1 **a** Different shapes of two characters. **b** Horizontal ligatures. **c** Vertical ligatures

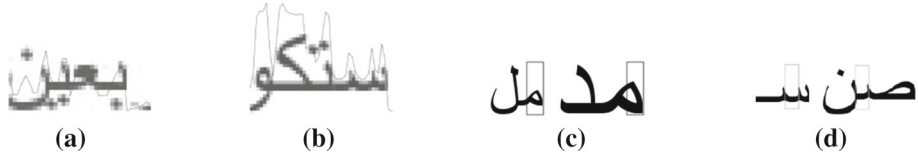


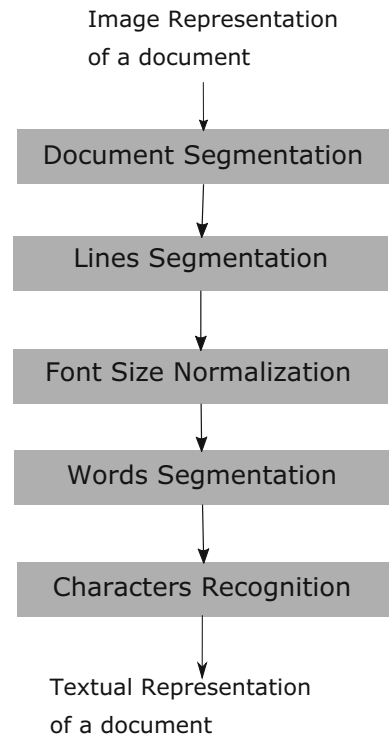
Fig. 2 **a** Threshold over-segmentation in left character. **b** Threshold over-segmentation in the right character, and under-segmentation in the middle. **c** Character context and scale. **d** Small window over-segmentation effect

handwritten text, Arabic and Farsi are cursive by nature. Thus, the typesetting is normally cursive in both machine generated and handwritten text. Segmentation of words to their constituting characters is a crucial step to the succeeding recognition phase. An Arabic character can have up to four different shapes according to its location in the word: isolated, start, middle, and end as shown in Fig. 1a. Some characters may only differ in the number of diacritics. In addition, characters may have different heights and widths. Moreover, some combinations of certain characters can have special ligatures to connect them as indicated in Fig. 1b, c. Due to these characteristics, segmentation algorithms can fall short by performing over-segmentation of wide characters, or under-segmentation of interleaving characters.

Since Arabic is cursive, then the range of research in the topic can be classified according to how the system recognizes words or sub-words. In [2–5], word level features are recognized to be identified as a word in a certain vocabulary set. On the other hand, works in [6,7] recognize characters features by using a preprocessing step to segment the input word into characters, then the segmented characters are recognized by a character recognition model. However, methods in [8,9] use a sliding window to recognize characters features.

In this paper, different sub-systems are proposed to solve some of the aforementioned complications. Our pipeline is an open-vocabulary Arabic text recognition system, which can recognize words that it was not trained on before. It starts by predicting the font size of the text, to normalize it to an 18pt font size that we use for the next steps. Then, a model is used for segmenting the word to its comprising characters, followed by a character recognition system. Many algorithms devised for segmentation of cursive Arabic documents, made use of the structural pattern of lower pixels density between characters. Based on this pattern, a histogram of horizontal projection has been widely used for segmentation [10,11]. However, this method is prone to over-segmentation when a character is composed of several ligatures, or under-segmentation due to the overlapping of characters (Fig. 2a, b). Other structural segmentation algorithms [6,7] depend on thinning or contour tracing to extract strokes or angles and use them as features for extraction. This method can solve the under-segmentation problem resulting from overlapping but is prone to over-segmentation. The characters recognition step is rather straightforward to accomplish, using decision trees [12], extracted moments [13], or other features as an input to a neural network. In [8,9,14,15], Hidden Markov Models (HMM) are used to learn character features and perform implicit segmentation.

Artificial Neural Networks have been showing state-of-the-art performance in object and characters recognition tasks [16,17]. They can learn hierarchical representations by stacking

Fig. 3 The flow diagram of the proposed Arabic OCR system

layers that learn on input features from the output of the previous layer. In the proposed system, a shallow neural network was built for the task of recognizing text font size. Another multichannel neural network [18] is used to predict the likelihood of a window, sliding on a sub-word image, of being a potential candidate cut place for characters segmentation. Afterwards, a convolutional neural network is applied to recognize the segmented characters. Convolutional layers [19] are employed to extract and learn features from the input image. Regularization techniques such as training data augmentation and dropout [20], are used to enhance network generalization by reducing the possibility of over-fitting.

The main contribution of this work is presenting a neural network pipeline (Fig. 3) consisting of three neural networks. The first network recognizes the font size of an input word. Then, the words can be normalized to 18pt font size, which will be the size of the characters that will be used to train the other two models. This font size prediction model has achieved a 99.1% test accuracy. A three-window segmentation neural network that is employed to segment a word to characters. The segmented output is then fed as input to a characters recognition model for finally recognizing the characters of a word. The proposed segmentation model can learn to explicitly segment Arabic words, of one font or preferably multiple fonts, into separated characters. The segmentation problem is formulated as a non-linear regression problem, assigning a sliding window values between $[-0.5, 0.5]$ indicating confidence in segmentation area, where positive values will indicate potential segmentation points while negative ones will occur inside characters as depicted in Fig. 12. It should be noted that the model at the segmentation step is totally blind to the character class, as it only recognizes the features of segmentation windows. The subsequent convolutional neural network will provide the character recognition step by classifying the segmented character into one of the

Arabic characters. The segmentation model produced a 98.9% accuracy, and the character recognition model achieved 94.38% accuracy when tested on the APTI dataset subset.

The rest of the paper is organized as follows. The preprocessing and the font size recognition step is described in Sect. 2. Then the segmentation model and its components are introduced in Sect. 3. The character recognition model is explained in Sect. 4. The details of the experiments carried out to test the three models are outlined in Sect. 5.

2 Document Preprocessing

For an Optical Characters Recognition task, there are many preprocessing and normalization steps required according to the type of models used to recognize the text. Many research projects were carried out in document layout analysis [21], extracting graphs and tables, binarization, segmentation, and alignment correction.

2.1 Document Segmentation

Segmentation of Arabic documents into basic constituents is an essential part of an OCR system. In the proposed system, an input document is segmented into lines (Fig. 4) and then into sub-words (Fig. 5) using the histogram approach for segmentation. Afterwards, a multi-layered neural network is used for recognizing the font size of a sub-word. Since the



Fig. 4 Vertical histogram projection of Arabic text lines

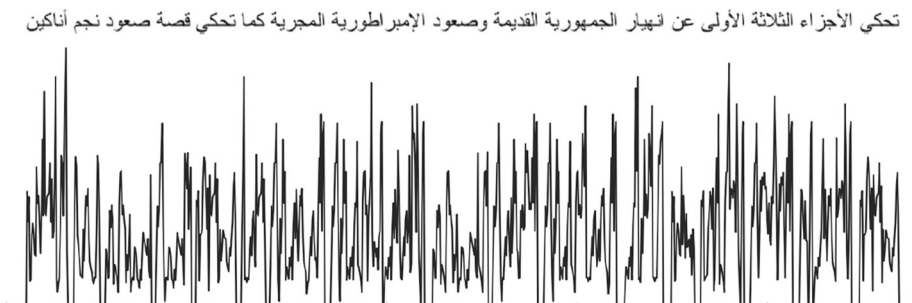


Fig. 5 Horizontal histogram projection of Arabic text line

proposed OCR model depends on recognizing the characters of specific font size, the font size of the word has to be identified.

2.2 Font Size Recognition Model

To develop multi-font Arabic OCR system, a model is developed to automatically recognize the font size of an input sub-word. Then the size information will be used to normalize the words into a default font size, 18pt. The rest of the proposed character recognition pipeline sub-systems is trained on words with this default font size to recognize input text.

2.3 A Neural Network for Font Size Recognition

The model is based on a neural network optimized specially for this task. Here, the input sub-word is first converted to a vertical histogram projection of the sub-word using Eq. 1 which is used as the input to the network.

$$His[j] = \sum_{i=0}^{W-1} word[i, j] \quad (1)$$

where $j \in [0, H]$ where H is the height of the word, W is its width, and $word[i, j]$ is the pixel (i, j) in the word. This representation helps the model to easily learn the height characteristics of each font size, specifically:

- Line height A : maximum height of a line.
- Ascender height $C + B$: the height of the line above the baseline.
- Base height C : the height of line's most dense area, which is around the baseline.
- Descender height $C + D$: the height of the line below the baseline.

The parameters A , B , C , and D are illustrated in Fig. 6. Figure 7 shows the vertical projection of words in three different sizes: 12, 18, and 24. It shows that the features used in our model can discriminate between different font sizes.

For font size detection, a three-layer neural network was employed. The neural network is trained to predict sizes used in APTI dataset which are: {6, 8, 10, 12, 18, 24}. The input layer size is determined by the maximum character height which is 40px for 24pt font size. So the input layer contains 40 units, and inputs smaller than 40 are padded with zeros. The hidden layer consists of fully interconnected nodes that transform the input to another representation for learning the output function that predicts the font size of the sub-word. The output layer contains 6 output units which are the number of font sizes we want to predict. The hidden layer is responsible for learning an intermediate representation of the input histogram. Experiments were conducted with both hyperbolic tangent function, and sigmoid function. The hyperbolic tangent function has an output domain in the range $[-1, 1]$, and zero mean. The sigmoid function output domain is $[0, 1]$ and 0.5 mean. The hidden layer units are a sigmoid squashing function while the output layer is composed of soft-max nodes for classification. This setup proved to give better results than tanh or sigmoid instead of soft-max function during experiments.

The model is trained on words generated from the set of sizes mentioned above, a held subset is used for validation, and a set from APTI dataset is used for testing.

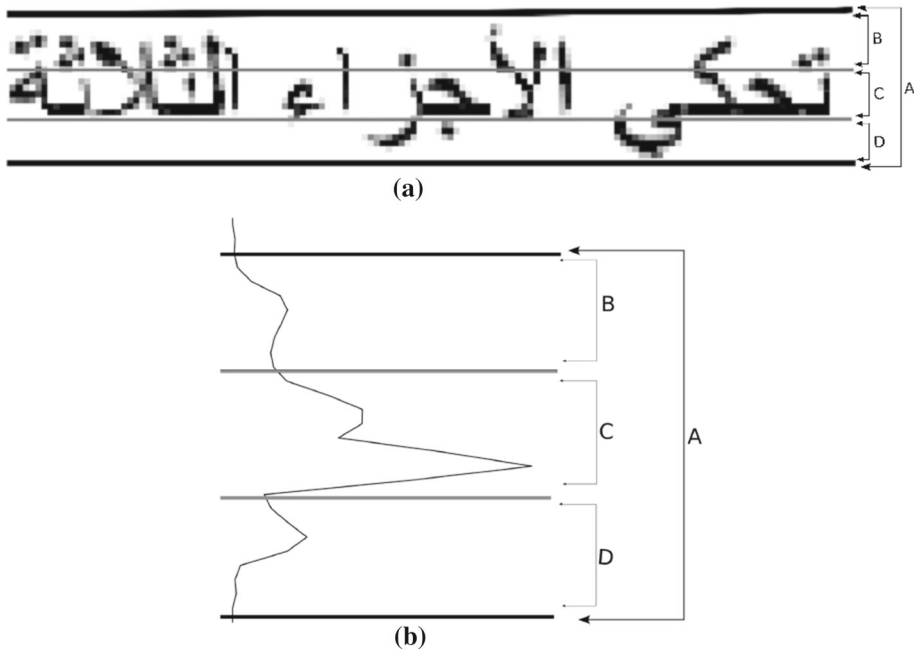


Fig. 6 Arabic font height properties. A is the line height. C is the base height. $C + B$ is the ascender height. $C + D$ is the descender height. **a** An example of an Arabic line text. **b** The vertical histogram projection of the text line in **(a)**

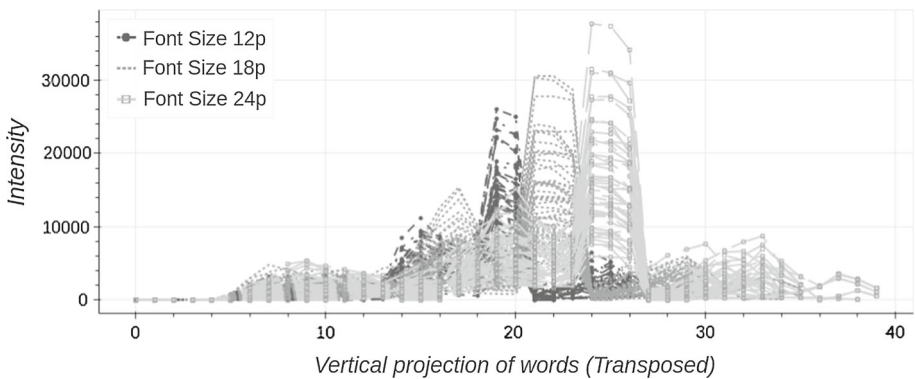


Fig. 7 Vertical histogram projection of Arabic words of three different sizes

3 The Segmentation Model

After the initial preprocessing steps, the document is segmented into lines and then sub-words. Afterwards words are normalized to the 18pt font size. The word or sub-word segmentation task, is to establish boundaries in-between characters of a word in the image. Applying a window to scan the word, a neural network is employed to predict the likelihood of the current window being in a segmentation area.

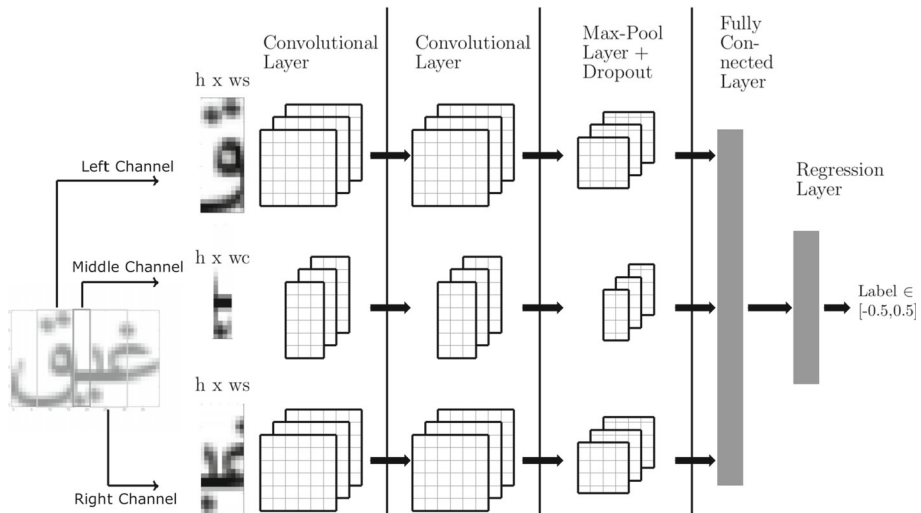


Fig. 8 Multichannel neural network for Arabic segmentation

Naturally, the difference in characters dimensions adds some uncertainty to how characters appear inside a window (Fig. 2c). To tackle this problem, data augmentation techniques (Sect. 5.2) are used to synthesize more training examples by applying some deformations on the training set [22]. Examples of transformations applied are scaling down or up, and translation; thus simulating what might exist in the test sets.

To solve the over-segmentation problem, a wider window is needed to recognize wide characters correctly (Fig. 2d). Nonetheless, increasing the window too much will add information other than the location of the segmentation, thus losing the cut localization, causing under-segmentation.

Based on the above segmentation remarks, two objectives need to be fulfilled:

1. To have as much context as possible inside a window without losing the segmentation localization.
2. To maintain a design that allows data augmentation without losing label consistency.

Using a neural network with a single wide window will fail the first point but satisfy the second. A Recurrent neural network for sequence labeling of every horizontal point in the window, resulted in a more complex network and impractical training times than what we developed later. The recurrent neural network design matrix does not allow for data augmentation as any change done to the input will make the labeled data incorrect.

A model that uses three windows as an input to a multichannel neural network is developed here (Fig. 8). In addition to using a small window, the middle window, for segmentation, two more channels are added to include the previous -right- window and the next -left- window. This increases the network input context. Transformations and deformations for data augmentation are applied to each window separately, without affecting the label consistency, hence satisfying the second point as well.

3.1 The Convolutional Channels

In the proposed architecture, each channel has separate two convolutional layers [19] that learn low level features. A convolutional layer contains a set of feature maps that convolves

an input feature $Y_{ijk}^{(l)}$ with a filter $w_i^{(l)}$ of size $A \times B$, where the component j, k of feature map i in l -th convolutional layer is given by:

$$H_{ijk}^{(l)} = B_i^{(l)} + \sum_{i'=1}^m \sum_{a=0}^A \sum_{b=0}^B w_{iab}^{(l)} * Y_{i'(j+a)(k+b)}^{(l-1)} \quad (2)$$

$$Y_{ijk}^{(l)} = \max \left(0, H_{ijk}^{(l)} \right) \quad (3)$$

where $\max(0, x)$ is a rectified linear activation function [23], and $B_i^{(l)}$ is a bias term. The number of feature maps in the previous layer is m , and $Y_{i'}^{(l-1)}$ is the output of the feature map i' from the previous layer. The two convolutional layers are followed by a max pooling layer to sub-sample the max value of every 2×2 window. A dropout technique is used to reduce over-fitting of data that is not complex enough. Dropping a percentage of neurons and their connections during training was found to increase the regularization of convolutional neural networks [20].

3.2 The Fully Connected Layer

A fully connected layer, that connects the output of the three channels, learns correlations between patterns that appear in each channel separately. The left channel should learn the right parts of the characters, while the right channel learns the left parts of characters, and the middle learns the area in-between. Figure 8 illustrates the channels' input and their output to the fully connected layer. A dense fully connected layer then learns the associations between each channel's specific features to find the correlation between the three of them. The layer is a linear function of its inputs as shown in Eq. 4, where the input from each channel is multiplied by its associated weights and summed with other channels and bias.

$$Y_i^{(h)} = \max \left(0, W_M Y_M + W_L Y_L + W_R Y_R + B_i^{(h)} \right) \quad (4)$$

where Y_x, W_x are the output of a channel and the weight associated with it for each $x \in \{L, M, R\}$ the left, middle, and right channels respectively. Left and right channels weights can be thought of as a bias that this layer learns for middle channel features. The last layer acts as regression layer that learns the likelihood of the window to be a segmentation place. This architecture is similar to multi-modal neural networks [18].

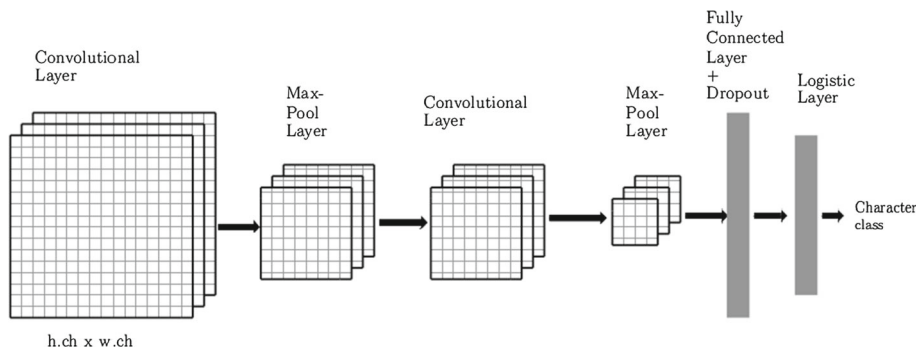
In the experiments, two models are trained on two different sets. First model was trained on one font, while the second model was trained on other four fonts. Models parameters are listed in Table 1. Stochastic gradient descent with 5×10^{-4} learning rate is used to minimize their mean squared error loss function.

4 The Character Recognition Model

The last neural network in the proposed system receives a segment of a word containing a character as an input. For better generalization and tolerance to shifts and errors resulting from segmentation model, training data will be augmented as discussed in Sect. 5.2. The character recognition model is a convolutional neural network (Fig. 9), with two convolutional layers and two max pooling layers, one for each convolutional layer. The convolutional layers apply the same rule as in Eq. (2). The first convolutional layer has 64 filters, and the second convolutional layer has 32 filters and both have 2×2 pooling layers. A fully connected

Table 1 Segmentation model parameters

	One-font model	Four-font model
Training fonts	1	4
Left and right channel filters	48, 48	64, 64
Middle channel filters	24	32
Convolutional window size	3	3
pool window size	2	2
Dropout	0.25	0.25
Fully connected layer size	180	256

**Fig. 9** Arabic character recognition neural network

layer (Eq. 4) with 64 neurons, followed by 25% neurons dropout, is used after the second convolutional layer. The output layer is a logistic regression layer that is finally used for classifying the input into an Arabic character. The neural network was trained by stochastic gradient descent, with 1×10^{-4} learning rate.

5 Experimental Results

In these experiments, it is assumed that the fonts to be recognized are known beforehand. Font size predictions are performed for six sizes: {6, 8, 10, 12, 18, 24}. Afterwards experiments are conducted on two 3-window segmentation models, one trained on one font and another on four fonts. Then the 3-window model and the 1-window model are compared on a common test set. Finally, a character recognition model is trained and then tested along with the segmentation model on APTI dataset.

5.1 Font Size Prediction Model

The font size prediction model is a neural network with one hidden layer. The experiments are conducted for different number of hidden nodes: {10, 50, 80, 120, 180, 200, 230, 270}. The activation function was also selected according to experiments carried out different types of activation functions for the hidden layer and logistic layer. A stochastic gradient

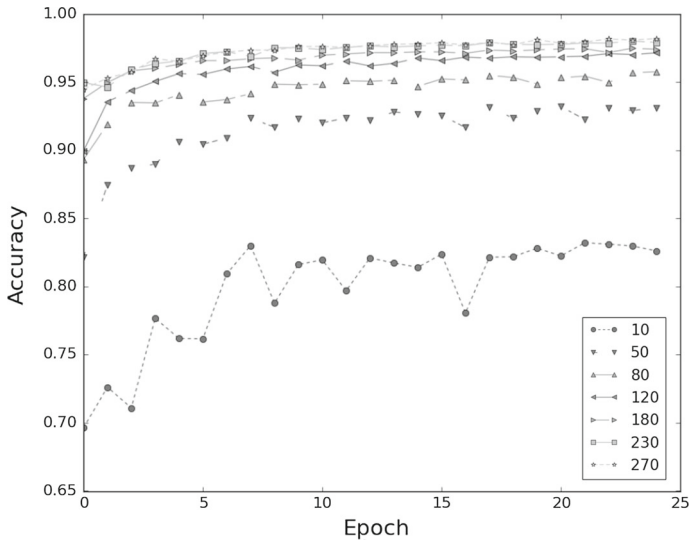


Fig. 10 Experiments results with tanh units in hidden and output layers. Each line represents the classification accuracy for a given hidden nodes size

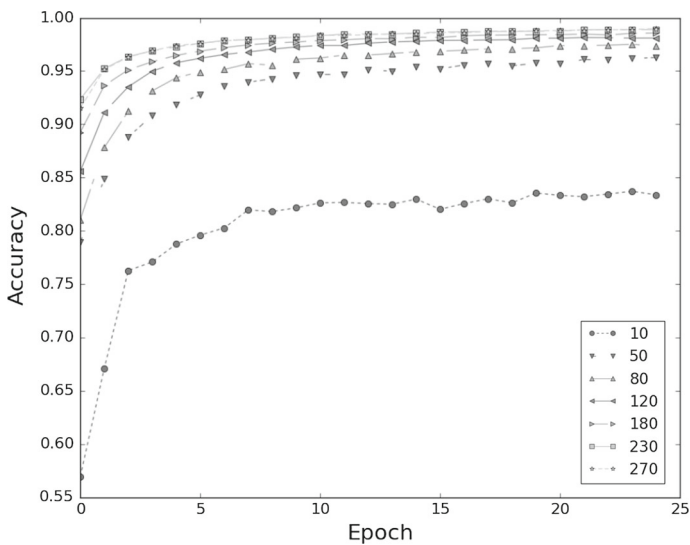


Fig. 11 Experiments results with sigmoid units in hidden and output layer. Each line represents the classification accuracy for a given hidden nodes size

descent learning algorithm was used to train the network on the classification task described in Sect. 2.3.

The neural network with sigmoid hidden activation units (Fig. 11) was found to converge better than the a network with hyperbolic tangent hidden activation units (Fig. 10). The best neural network with sigmoid hidden units achieved 99.1% test accuracy, while the neural network with tanh hidden units achieved 97.2% test accuracy.

5.2 Data Augmentation

To improve generalization capability of the model and to reduce over-fitting, the training data size is increased by applying signal transformations to the given images. These transformations are selected so as to preserve the input label, and add desired invariances to the model. For two dimensional objects recognition such as in recognizing handwritten digits from MNIST, it was shown that augmenting the training dataset resulted in significance accuracy improvement [22]. For our problem, slight translation invariance was added on horizontal and vertical dimensions which should add tolerance to segmentation variability. Rotation and shear invariance with small angles were also used, which is useful for italic variations. In addition, zoom in and out transformations were applied to account for text size changes.

5.3 The Segmentation Model

For segmentation model experiments,¹ a set of defined fonts were selected for training and testing; and a constant font size of 18pt for generating training data was used. For testing the segmentation model, two instances of our proposed model were trained, the first was trained on one font, while the second was trained on the other four fonts. The right and left windows were of width 12px which is the mean width of Arabic characters at size 18pt, whereas the middle window width was 4px. The three windows has 26px height which is the maximum height of characters at that size. Another model with a single window was also tried for comparison, where the single window was of width 12px.

For testing, an Arabic dictionary dataset² was used to establish two datasets for evaluation; the first contains words written in one font and the other in the other four fonts.

5.3.1 Data Generation

Font files are utilized to generate random words to make up the training data sets. Given a textual word we generate an image as a training example, hence we are able to get the segmentation places between characters. The ground truth of a word segmentation (Fig. 12) can be found in two steps:

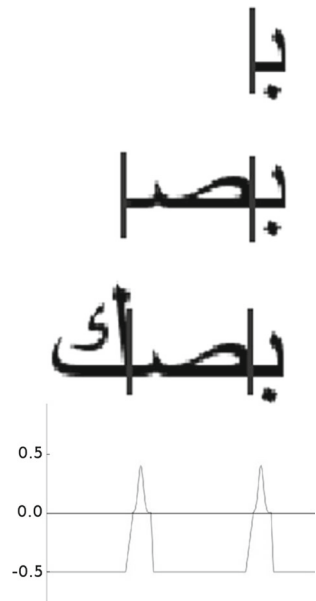
1. Characters are drawn sequentially and the width is measured each time to find the ideal segmentation (Fig. 12).
2. For each segmentation place, we assign the label of the segmentation pixel and its corresponding pixels using a Gaussian function, that has a mean at the center pixel from first step.

Examples are generated by sliding a window, of width 4px and height 26px, on the word. One example is added for each window step, consisting of: the sliding window as the middle window, in addition to the windows to its left and right. Each example will be added as a negative sample (Fig. 13b) if it is inside a character's boundaries. On the contrary, if the middle window starts in a segmentation place, then it is added as a positive sample, and its probability is given by the Gaussian function associated with this segmentation area found in step 2 above (Fig. 13c). This algorithm is repeated for 2100 words, each word i having n_i random characters where $n_i \in \{2, 3, 4, 5\}$. One third of these words are held out as a validation data set that is used for parameters selection.

¹ Keras was used for experiments <https://github.com/fchollet/keras/>.

² Data used from <https://sites.google.com/site/motazsite/>.

Fig. 12 The method used for finding the ground truth of segmentation for three letter word, by drawing them iteratively and using each width as a valid segmentation. Below is the function used for assigning labels to the given example



Data augmentation technique, described in Sect. 5.2, is used on the training data subset. For each training example, three of these deformations are selected randomly to apply on each window. The resulting new three windows are added as a new example with the same label. The size of data with labels greater than zero (segmentation areas) are increased explicitly to balance with the data of labels less than zero, to ensure classes balance during training. Augmentation increases the training data and leads to better generalization on the test set [22].

The first segmentation model uses Arial font in the generation. The second model uses the following four fonts: Arial, Tahoma, Thuluth, and Damas, which are very different typographically. The test sets are generated similarly using about 250,000 real Arabic words from a dictionary. These words range in length from two to six characters. Two test sets are generated using this dictionary, first using the Arial font, and the other test set is generated using the same previous four fonts.

5.3.2 Segmentation Model Results

To evaluate the segmentation model, a balanced data set is used to measure the mean squared error, which is used as a loss function. The regression problem is converted to a classification problem by treating the model output that's greater than 0.2 as a true segmentation and less than 0.2 as negative segmentation. The same test data set is used to measure the classification accuracy.

For each trained model, Table 2 shows the loss function values result for training, validation, and two test sets. In addition, the classification accuracy is reported for the training and test sets in Table 2. The 1-Font model is able to segment 4-font test set with accuracy of 65.5%, and thus being able to get around 40% of the other fonts' segments right.

The 3-window model accuracy and experimental results from other segmentation models [1], are listed in Table 3. While the datasets used in all other experiments were not open

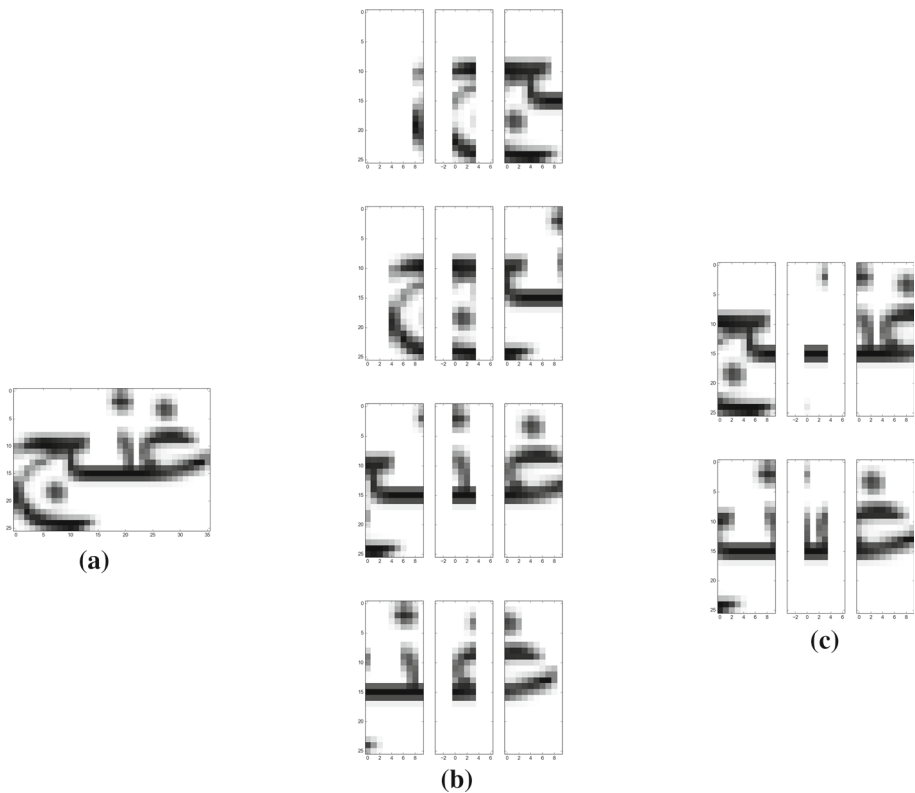


Fig. 13 **a** The data for this word is extracted by a sliding window. **b** Middle windows that are over a character are recognized as negative samples and added to the data set. **c** Middle windows that are over a segmentation place are added as a positive samples to the data set

Table 2 The loss function values on the regression task and the accuracy on the classification test set

	Recall Loss	Validation Loss	1-Font test Loss	Accuracy (%)	4-Font test Loss	Accuracy (%)
1-Font 3-window model	0.0247	0.0315	0.0335	98.9	0.110	65.8
4-Font 3-window model	0.0301	0.0309	0.0431	98.7	0.0465	95.5

and listed for reference, our model is compared directly to a 1-window mode trained on one font data. Both networks are tested with the same 1-font test data. The 1-window model has produced 90.2% accuracy that was outperformed by the 3-window model with its vigilance towards confusing windows that cause under and over segmentation.

Figure 14a shows the correct segmentation of a sample document containing previously over-segmented and under-segmented characters being correctly segmented by the proposed model. The data examples that maximally activates neurons in the left channel are shown in (Fig. 14b), and the right channel shown in (Fig. 14c). The windows that maximally activate the left channel are mostly right parts of characters that appear next to a cut. Similarly, The right channel learns left parts of the character that appear right to a cut place.

Table 3 Subjective comparison results of segmentation methods

	Description	Data	Accuracy
Zidouri and Abdelmalek [24]	Structural Method, that extracts engineered features and apply rules to segment characters Independent of font size and font type	200 images	90%
Broumandnia et al. [25]	Based on wavelet transform to solve segmentation Extracted wavelet coefficients are exploited, in detecting, underlying horizontal edges and base line Rules are used to find valid segmentation points	1000 words different sizes and fonts	97.83%
Nawaz et al. [13]	Vertical and horizontal projection	Many document images each containing about 200 characters	76%
Bushofa and Spann [26]	Extracts contour information Chooses the right position of segmentation which is indicated by an angle formed between each pair of joined characters	1065 characters from each font are tested	97.01%
Hamid and Haraty [27]	Structural features for feed-forward multilayer neural networks Generates presegmentation points for these blocks by oversegmentation. A neural network is subsequently used to verify the accuracy of these segmentation points	10,000 exemplars	69.72%
Touj et al. [15]	Uses HMM with standard Hough transform as method 1 and with generalized Hough transform as method 2 Using a dynamic sliding window that starts from left to right to find characters boundaries	6400 characters	Method 1: 91%, and Method 2: 97%
1-Font one window model	Neural Network similar to the proposed model but with only one window Sliding windows calculates the probblity of segmentation points inside one window	250,000 Words	90.2%
1-Font 3-windows model	Our proposed multi-channel neural network A sliding window with another two channels are used as a context to calculate the probability of segmentation point	250,000 words	98.9%

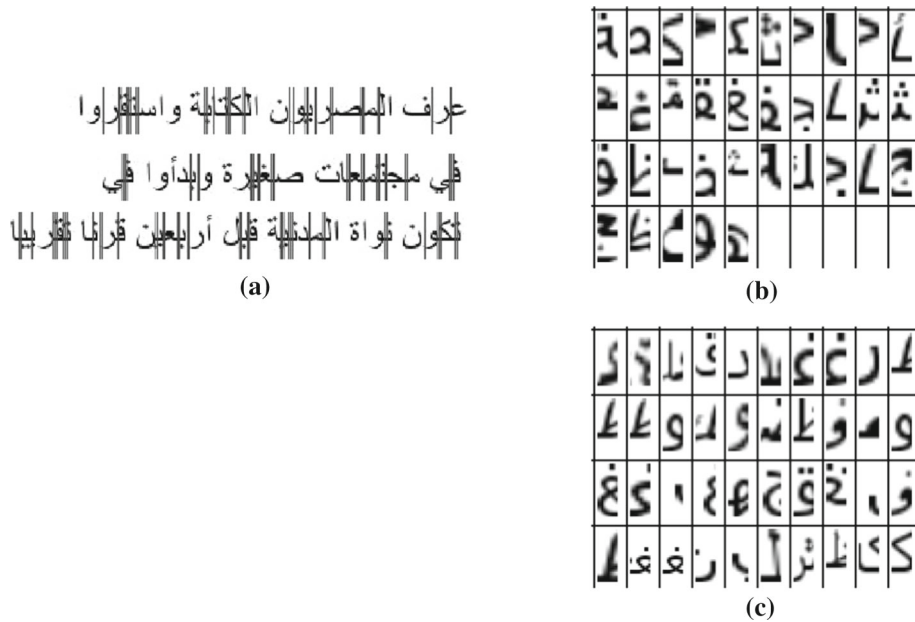


Fig. 14 **a** Segmentation of a document using the model. **b** Segments of words that have maximal neural response in the left channel, and the right channel in (c)

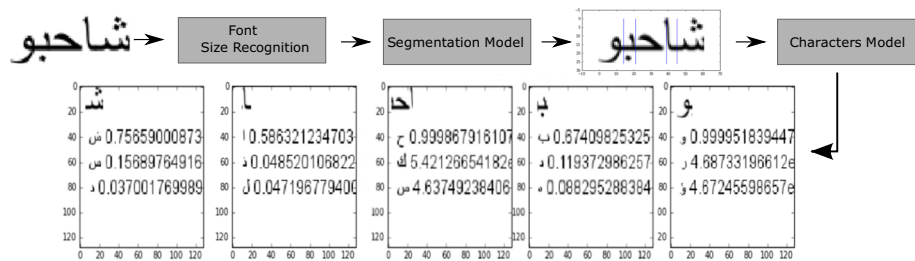


Fig. 15 Pipeline for APTI data set

5.4 The Character Recognition Model

A convolutional neural network is used for characters recognition. For training data, a data generation technique similar to segmentation model data is used. Examples for each Arabic characters are generated in different contexts: isolated, to the right, to the left, and in the middle. The set is split to 66% for the training set and the remaining for the validation set. The training set is used to generate more examples as described in Sect. 5.2.

5.4.1 Evaluation on APTI Data Set

Arabic Printed Text Image (APTI) data set is a publicly available large scale benchmark for recognition systems in Arabic. The database contains low-resolution (72 DPI) images. It consists of synthetically generated printed Arabic word images in many fonts, sizes, and styles.

Table 4 Comparison with Arabic recognition systems evaluated using the APTI database

OCR Systems	WER (%)	CER (%)	Description
IPSARec System [28]	22.5	3.2	Based on Discrete HMM from Hidden Markov Model Toolkit (HTK) Omnifont Unlimited vocabulary recognition system It does not require segmentation Pixel density features from the text image and its horizontal and vertical derivatives
UPV-PRHLT [28]	15.6	4.0	Based on Bernoulli HMMs (BHMMs) Trained from input images scaled in height to 40 pixels One model for each font size is trained
Siemens [29]	0.0	0.06	Based on recurrent LSTM neural networks Each layer is recurrent getting its input not only from the input pixel, but also from the neighboring cells within the layer A different neural network is trained for each font size Depending on test image size, one of multiple trained networks is chosen for recognition
THOCR1 [29]	8.23	1.05	HMM-based OCR system Statistical and structural features and their derivatives
THOCR2 [29]	4.97	0.81	HMM-based OCR system Statistical and structural features and their derivatives Four-gram language model trained on the APTI training corpus used for rescoring
Proposed pipeline	5.62	1.37	A neural network for recognizing font size Input images are then normalized to single font size 18 pt

Table 4 continued

OCR Systems	WER (%)	CER (%)	Description
			A multichannel neural network segments the word into characters
			A characters recognition model is used to recognize words

Table 5 CER and WER results for the Arabic Transparent font in each size

OCR system/size		6	8	10	12	18	24	Mean
IPSAR	WER	94.3	26.7	25.0	16.9	22.9	22.5	34.7
	CER	40.6	5.8	4.9	3.1	4.3	3.2	10.3
UPV-BHMM	WER	2.8	0.3	0.2	0.1	0.1	0.1	0.6
	CER	0.5	0.1	0.1	0.0	0.0	0.0	0.1
Siemens	WER	0.1	0.1	0.0	0.1	0.0	0.0	0.1
	CER	0.0	0.0	0.0	0.0	0.0	0.0	0.0
THOCR	WER	10.5	4.2	5.2	7.5	5.4	5.0	6.3
	CER	1.7	0.5	0.8	0.9	0.9	0.8	0.9
Proposed pipeline	WER	11.5	8.9	4.6	3.8	2.4	2.3	5.6
	CER	3.5	1.9	1.1	0.9	0.2	0.2	1.3

The subset of Arabic Transparent Font is used as a test set to test the performance of the pipeline of the three models: Fonts size recognition model, Segmentation model and Characters recognition model. The input to pipeline from APTI dataset is an Arabic word (example in Fig. 15). The word font size is first recognized and then normalized to 18pt size. Then, the word is segmented into characters using the Segmentation model. Finally, each character is recognized by the character recognition model.

The pipeline was run on the subset of Arabic Transparent APTI dataset and the result is reported in Table 4. The Arabic Transparent font is used in this comparison because this font was included in the experimental protocol APTIPC1 for the text recognition competitions that used the APTI database [28,29]. The database has five sets. According to the experimental protocol, four sets are used for training and the fifth set is used for testing. To measure the accuracy rate of the OCR system, the word error rate (WER) which is the count of misclassified samples is used. In addition, the character error rate (CER) is used which considers the errors due to insertion, deletion, and substitution.

The proposed pipeline OCR system reported 94.38% accuracy (5.62% WER and 1.37% CER error rates). However, some of the OCR systems reported in this comparison used word lexicons and n -gram language models. That leaves a room for more improvement on the proposed character recognition model we intend to do in our future work.

ICDAR 2011 and ICDAR 2013 competitions reported the recognition results on multi-size Arabic Transparent font. Table 5 compares the results of the proposed system and the OCR systems results on different sizes of Arabic Transparent font, reported in these competitions.

6 Conclusion

A system for Arabic OCR was developed using three neural networks for three different tasks: font normalization, word segmentation, and characters recognition. A neural network was used to recognize the font size of a word by using the word's vertical histogram projection as a feature and the model reported a test accuracy of 99.1%. A multichannel neural network is used to segment the word into characters. It incorporates a sliding window as a middle channel and another next and previous windows for additional context. This model performs quite well against over-segmentation and under-segmentation problems. A model trained to segment one Arabic font has an accuracy of 98.9% while a model trained to segment four fonts has a 95.5% accuracy. This model has proved to outperform the one-window model in test accuracy. The segmentation model output is then employed to recognize the characters using a convolutional character recognition model. This proposed pipeline was tested against an APTI dataset subset showing 94.38% accuracy. Future work would be directed to improve the character recognition process in the pipeline. The pipeline will be useful for automatically building an open vocabulary Arabic OCR system, as the generation and training process can become autonomous with very few variables or features to be tuned.

References

1. Alginahi YM (2013) A survey on Arabic character segmentation. *Int J Doc Anal Recognit* 16(2):105–126
2. AlKhateeb JH, Khelifi F, Jiang J, Ipson SS (2009) A new approach for off-line handwritten Arabic word recognition using knn classifier. In: 2009 IEEE international conference on signal and image processing applications (ICSIPA), pp 191–194
3. Al-Badr B, Haralick RM (1995) Segmentation-free word recognition with application to Arabic. In: *Proceedings of the third international conference on document analysis and recognition*, vol 1, pp 355–359
4. Khorsheed MS, Clocksin WF (2000) Multi-font Arabic word recognition using spectral features. In: *Proceedings of 15th international conference on pattern recognition*, vol 4, pp 543–546
5. Jelodar MS, Fadaeieslam MJ, Mozayani N, Fazeli M (2005) A Persian OCR system using morphological operators. In: *World Academy of Science, Engineering and Technology*, pp 137–140
6. Märgner V (1992) Sarat-a system for the recognition of Arabic printed text. In: *Proceedings of 11th IAPR international conference on pattern recognition 1992*, vol II. Conference B: pattern recognition methodology and systems, pp 561–564
7. Azmi R, Kabir E (2001) A new segmentation technique for omnifont farsi text. *Pattern Recognit Lett* 22(2):97–104
8. Khoury I, Giménez A, Juan A, Andrés-Ferrer J (2015) Window repositioning for printed Arabic recognition. *Pattern Recognit Lett* 51:86–93
9. Ahmad I, Mahmoud SA, Fink GA (2016) Open-vocabulary recognition of machine-printed Arabic text using hidden markov models. *Pattern Recognit* 51:97–111
10. Amin A (1998) Off-line Arabic character recognition: the state of the art. *Pattern Recognit* 31(5):517–530
11. Zheng L, Hassin AH, Tang X (2004) A new algorithm for machine printed Arabic character segmentation. *Pattern Recognit Lett* 25(15):1723–1729
12. Bushofa BMF, Spann M (1997) Segmentation and recognition of printed Arabic characters using structural classification. *Image Vis Comput* 15(3):167–179
13. Nawaz SN, Sarfraz M, Zidouri A, Al-Khatib WG (2003) An approach to offline Arabic character recognition using neural networks. In: *Proceedings of the 2003 10th IEEE international conference on electronics, circuits and systems, 2003 (ICECS 2003)*, vol 3, pp 1328–1331
14. Gouda AM, Rashwan MA (2004) Segmentation of connected Arabic characters using hidden markov models. In: *2004 IEEE international conference on computational intelligence for measurement systems and applications (2004 CIMSAP)*, pp 115–119
15. Touj S, Amara NB, Amiri H (2007) Two approaches for Arabic script recognition-based segmentation using the Hough transform. In: *International conference on document analysis and recognition (ICDAR-2007)*, pp 654–658

16. Cireřan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J (2011) High-performance neural networks for visual object classification. Preprint [arXiv:1102.0183](https://arxiv.org/abs/1102.0183)
17. Lee C-Y, Xie S, Gallagher P, Zhang Z, Tu Z (2014) Deeply-supervised nets. Preprint [arXiv:1409.5185](https://arxiv.org/abs/1409.5185)
18. Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY (2011) Multimodal deep learning. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 689–696
19. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
20. Srivastava N (2013) Improving neural networks with dropout. Ph.D. thesis, University of Toronto
21. Cattoni R, Coianiz T, Messelodi S, Maria Modena C (1998) Geometric layout analysis techniques for document image understanding: a review. ITC-irst Technical Report 9703(09)
22. Simard PY, Steinkraus D, Platt JC (2003) Best practices for convolutional neural networks applied to visual document analysis. In: International conference on document analysis and recognition (ICDAR-2003), pp 958–963
23. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: International conference on artificial intelligence and statistics, pp 315–323
24. Zidouri A (2010) On multiple typeface Arabic script recognition. *Res J Appl Sci Eng Technol* 2(5):428–435
25. Broumandnia A, Shanbehzadeh J, Nourani M (2007) Segmentation of printed farsi/Arabic words. In: IEEE/ACS international conference on computer systems and applications 2007 (AICCSA'07), pp 761–766
26. Bushofa BMF, Spann M (1997) Segmentation of Arabic characters using their contour information. In: 13th International conference on digital signal processing proceedings, 1997 (DSP 97), vol 2, pp 683–686
27. Alaa H, Ramzi H (2001) A neuro-heuristic approach for segmenting handwritten Arabic text. In: ACS/IEEE international conference on computer systems and applications 2001, pp 110–113
28. Slimane F, Kanoun S, El Abed H, Alimi AM, Ingold R, Hennebert J (2011) ICDAR2011-Arabic recognition competition: multi-font multi-size digitally represented text. In: International conference on document analysis and recognition (ICDAR-2011), pp 1449–1453
29. Slimane F, Kanoun S, El Abed H, Alimi AM, Ingold R, Hennebert J (2013) ICDAR2013 competition on multi-font and multi-size digitally represented Arabic text. In: International conference on document analysis and recognition (ICDAR-2013), pp 1433–1437