Predictive Segmentation Using Multichannel Neural Networks in Arabic OCR System

Mohamed A. Radwan(✉), Mahmoud I. Khalil, and Hazem M. Abbas

Computers and Systems Engineering Department, Faculty of Engineering, Ain Shams University, Cairo, Egypt
mohamedat.Radwan@gmail.com

Abstract. This article offers an open vocabulary Arabic text recognition system using two neural networks, one for segmentation and another one for characters recognition. The problem of words segmentation in Arabic language, like many cursive languages, presents a challenge to the OCR systems. This paper presents a multichannel neural network to solve offline segmentation of machine-printed Arabic documents. The segmented characters are then used as input to a convolutional neural network for Arabic characters recognition. The accuracy of the segmentation model using one font is 98.9 %, while four-font model showed 95.5 % accuracy. The accuracy of characters recognition on Arabic Transparent font of size 18 pt from APTI data set is 94.8 %.

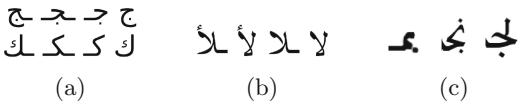
Keywords: Arabic segmentation · OCR · Convolutional neural networks

1 Introduction

In the classic topic of Arabic characters recognition, we are concerned about digitizing Arabic documents into electronic format. Since Arabic is cursive, so the range of research in the topic can be classified according to how the system recognizes words or sub-words. In [1–4], word level features are recognized to classify them into a word in a vocabulary set. On the other hand [5,6], recognize characters features by using a preprocessing step to segment the input word, then the segmented characters are recognized by a character recognition model. While [7,8], use a sliding window to recognize characters features.

This paper offers an open-vocabulary Arabic text recognition system using two neural networks, one for the segmentation and another one for characters recognition. Automatic segmentation of Arabic has always been a tough problem to solve [9]. Unlike Latin languages which are cursive mostly in handwritten text, Arabic and Farsi are cursive by nature, so typesetting is 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 placement in the word: isolated, start, middle, and end (Fig. 1a). Some characters may only differ

F. Schwenker et al. (Eds.): ANNPR 2016, LNAI 9896, pp. 233–245, 2016.
DOI: 10.1007/978-3-319-46182-3_20



(a) (b) (c)

Fig. 1. (a) Different shapes of two characters. (b–c) Horizontal and vertical ligatures.

in the number of diacritics. Characters also have different heights and widths. Defined combinations of certain characters can have special ligatures to connect them (Fig. 1b and c). Due to these characteristics, segmentation algorithms can fall short by over-segmentation of wide characters, or under-segmentation of interleaving characters.

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 (Figs. 2a, b). Other structural segmentation algorithms [5,6], depend on thinning or contour tracing to extract strokes or angles and use them as features for extraction. This way they can solve the under-segmentation resulting from overlapping but suffers from over-segmentation. For characters recognition stage the problem becomes much easier, [12] uses a decision tree, [13] uses extracted moments and other shape features as an input for a Neural Network. In [7,8,14,15] Hidden Markov Models are used to learn characters features and do implicit segmentation.

Recently Neural Networks are shown to have state-of-the-art results in object and characters recognition tasks [16,17]. They learn hierarchical representations by stacking layers that learn features in each one from the output of the previous layer. In this paper, a multichannel neural network [18] is used to predict the likelihood of a window, sliding on a sub-word image, being on candidate cut place for characters segmentation. Afterwards, another convolutional neural network is used to recognize the segmented Arabic characters. Advances in training deep neural networks are exploited to reduce over-fitting. Convolutional layers [19] are used for learning features from the image. Regularization techniques like training data augmentation and dropout [20], are used to enhance network generalization.

(a) (b) (c) (d)

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.

The contribution of this paper is a segmentation 3-windows neural network that is used in our OCR pipeline. The segmented output is then used as input to a characters recognition model. 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 (Fig. 4a). The model is blind to characters classes, as it only recognizes the features of segmentation windows. The character recognition model is formulated as a convolutional neural network that classifies a segmented character into one of the Arabic characters. The segmentation model established a 98.9 % accuracy and the character recognition model had 94.8 % accuracy against the APTI dataset subset.

The paper starts by introducing the segmentation model and its components in Sect. 2. The character recognition model is described in Sect. 3. The details of experiments carried out to test this model are listed in Sect. 4.

2 The Segmentation Model

The typical OCR system can easily segment a document into lines then words. The word or sub-word segmentation task, is to recognize boundaries in-between characters of a word, given its image. Using a scanning window on the word, the model predicts a likelihood of the current window to be a segmentation area. The variances due to the difference in characters dimensions add lots of aberrations to how characters appear inside a window (Fig. 2c). Data augmentation techniques (Sect. 4.1) are used to synthesize more data by applying some deformations on the training set [21]. The functions chosen are scaling down or up, and translation; so to simulate what might happen in test sets. To solve over-segmentation, 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, or under-segmentation. Taking into consideration these problems two rules are established:

1. Have as much context inside a window without losing the cut localization.
2. Maintain a design matrix that allows data augmentation without losing label consistency.

A neural network with a single wide window as input will fail the first point but allow the second. A Recurrent neural network for sequence labelling of every horizontal point in the window, aside from being hard to train, will fail the second point. A model that uses three windows as input to a multichannel neural network is developed (Fig. 3). So beside using a small window for segmentation, another two channels are added as a previous window and a next window. This will increase the network input context. Data augmentation is carried out by applying deformations to each window separately, without affecting the label consistency thus satisfying the second point as well.

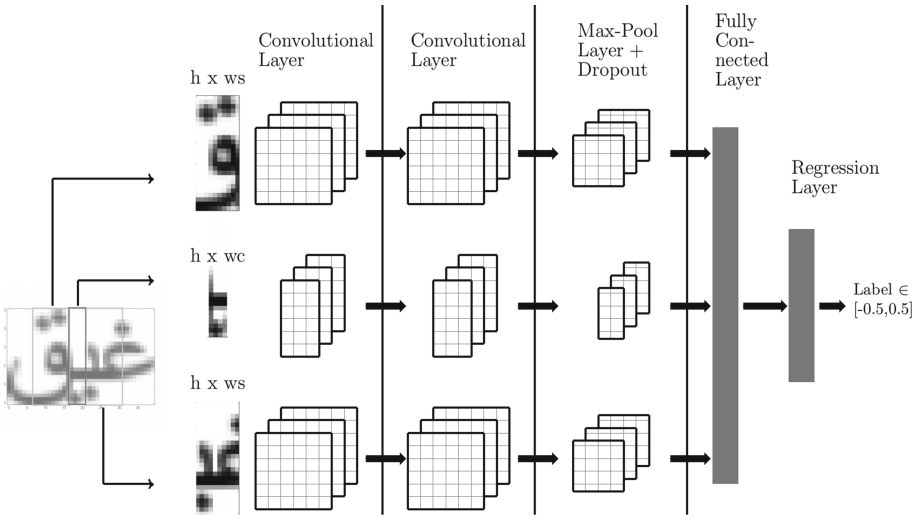


Fig. 3. Multichannel neural network for Arabic segmentation.

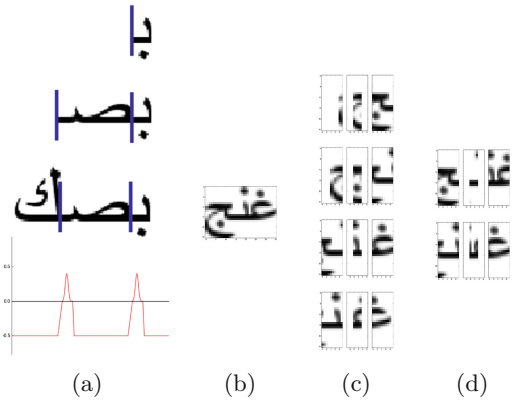


Fig. 4. (a) Finding the ground truth of segmentation, and the function used for assigning labels. (b) The data for this word is extracted by a sliding window. (c) Examples labelled as −0.5. (d) Examples labelled greater than 0.

2.1 The Convolutional Channels

Each channel learns low level features using convolutional layers [19]. A layer contains a set of features mapping input to output by convolving a filter $w_i^{(l)}$ of size $A \times B$. $Y_{ijk}^{(l)}$, the component j, k of feature map i in l-th convolutional layer is given by:

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

$$Y_{ijk}^{(l)} = max(0, H_{ijk}^{(l)}) \tag{2}$$

where $max(0, x)$ is a rectified linear activation function [22], $B_i^{(l)}$ is a bias term. The number of feature maps in the previous layer is m, $Y_{i'}^{(l-1)}$ is the output of the feature map i' from the previous layer. A max pooling layer is added after the convolutional layers that sub-samples the max value of every 2×2 window. Then, a dropout is used to reduce over-fitting of data that is not complex enough. It drops some neurons and their connections during training and found to increase the regularization of convolutional neural nets [20].

2.2 The Fully Connected Layer

The model's fully connected layer 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. A dense fully connected layer then learns the associations between each channel's specific features to find the correlation between the three of them.

$$Y_i^{(h)} = max(0, W_M Y_M + W_L Y_L + W_R Y_R + B_i^{(h)}) \tag{3}$$

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. At the end there is a regression layer that learns the likelihood of the window to be a segmentation place. This architecture is similar to multi-modal neural networks [18].

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

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

3 The Character Recognition Model

Our second model accepts a segment of a word containing a character as an input. The model training data is augmented as discussed below in Sect. 4.1 for better generalization and tolerance to shifts and errors resulting from segmentation model. It's based on convolutional neural network (Fig. 5), with two convolutional layers and two max pooling layers after each one. The convolutional layers are modelled as in Eq. (1). First convolutional layer has 64 filters, and second convolutional layer has 32 filters and both have 2×2 pooling layers. A fully connected layer (Eq. 3) with 64 outputs is used after the convolutional layers, it's then followed by 25 % dropout. The last layer is a logistic regression layer for classifying the input from the last layer into an Arabic character. Stochastic gradient descent, with $1 \cdot 10^{-4}$ learning rate, was also used as a learning algorithm.

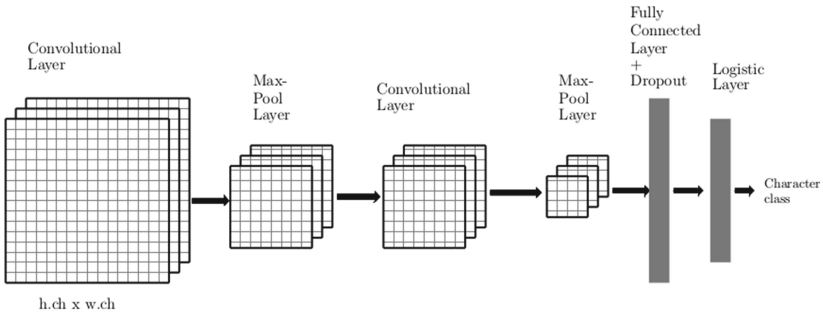


Fig. 5. Arabic character recognition neural network.

4 Experiments

In these experiments, we assume that the fonts to be recognized are known beforehand and that the input font size will be normalized to the font size we trained on. We conduct experiments on two 3-windows segmentation models, once trained on one font and another on 4 fonts. We then compare 3-windows model to 1-window model on common test set. Finally, A characters model is trained and then tested along with segmentation model on APTI dataset.

4.1 Data Augmentation

To improve generalization and reduce over-fitting we increase the training data by applying signal transformations to images. These transformations are selected so as to preserve input label, and add desired invariances to the model. For two dimensional objects recognition such as in recognizing handwritten digits from

MNIST, [21] proved that augmenting the training dataset resulted in significance accuracy improvement. For our problem, we added slight translation invariance 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 we used zoom in and out transformation to account for text size changes.

4.2 Segmentation Model

For segmentation model experiments[1], we used a set of defined fonts for training and testing; and a constant font size of 18pt for generating training data. For this size the mean characters width was found to be 12 pixels hence it was used afterwards as windows width, and the height for all windows was set to 26 pixels. To test the segmentation method, two instances of our proposed model were constructed, the first is trained on one font, while the second on four fonts. The right and left windows were of width 12 pixels, whereas the middle window width was 4 pixels. Another model with a single window was also constructed for comparison where the single window was of width 12 pixels.

For testing, we used an Arabic dictionary dataset[2]. Using this dataset, two test sets are constructed for evaluation; the first contains words written in one font and the other in four fonts.

Data Generation. The ground truth of a word segmentation (Fig. 4a) can be found in two steps:

1. Characters are drawn sequentially and width is measured each time to find the ideal segmentation (Fig. 4a).
2. For each segmentation place, we assign the label of the segmentation pixel and pixels around it by a Gaussian function, that has a mean at the center pixel from first step.

Then, examples are generated by sliding a window, of width 4 pixels and height 26 pixels, on the word. One example is added for each window step, consisting of: the sliding window as the middle window, plus the windows to its left and right. Each example will have label -0.5 (Fig. 4c) if it is inside a character's boundaries. Otherwise if the middle window starts in a segmentation place, then its label is given by the Gaussian function associated with this segmentation area found in step 2 above (Fig. 4d). This algorithm is repeated for 2100 words, each word having Wn random characters where $Wn \in \{2, 3, 4, 5\}$. One third of these words are held out as a validation data set that is used for parameters selection.

The data selected for the training set is used for augmentation procedure. We use data augmentation technique described in Sect. 4.1, on the training data subset. For each training example, three of these deformations are selected randomly to apply one on each window. The resulting new three windows are added

[1] Keras was used for experiments https://github.com/fchollet/keras/.
[2] Data used from https://sites.google.com/site/motazsite/.

as a new example with the same label. The size of data with labels greater than zero (segmentation areas) are increased explicitly to balance them with the data of labels less than zero, to ensure classes balance during training.

The first 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 yet 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 same previous four fonts.

4.3 The Characters Model

Using similar generation technique as in segmentation model data, training and validation sets are generated for characters model. For each Arabic character, we generate examples for it in different contexts that applies to it: alone, to the right, to the left, and in the middle. The set is split to 66 % for training set and the remaining for validation set. The training set is used to generate more examples as described in Sect. 4.1.

4.4 Evaluation on APTI Data Set

Arabic Printed Text Image (APTI) data set is a large scale benchmark for recognition systems in Arabic. A subset of Arabic Transparent Font of size 18 is used to text a pipeline of Segmentation system and Characters recognition system (Fig. 6).

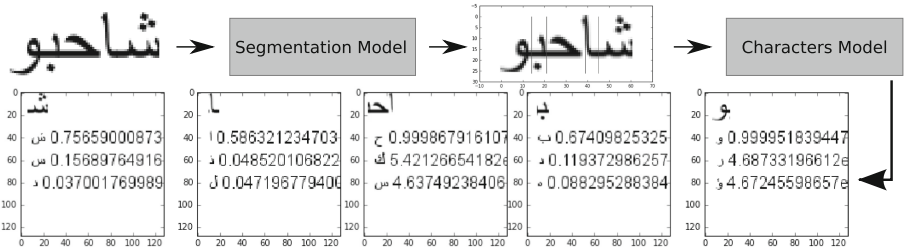


Fig. 6. Pipeline for APTI data set

4.5 Experimental Results

Segmentation Model. The loss function values of each trained model are listed in Table 2 for training, validation, and two test sets. Besides reporting the loss of the test sets, an accuracy is also reported. The test set is converted into classification by checking if the model would assign the segmentation ground truth (Fig. 4a) with labels greater than 0.2. This accuracy is reported on both

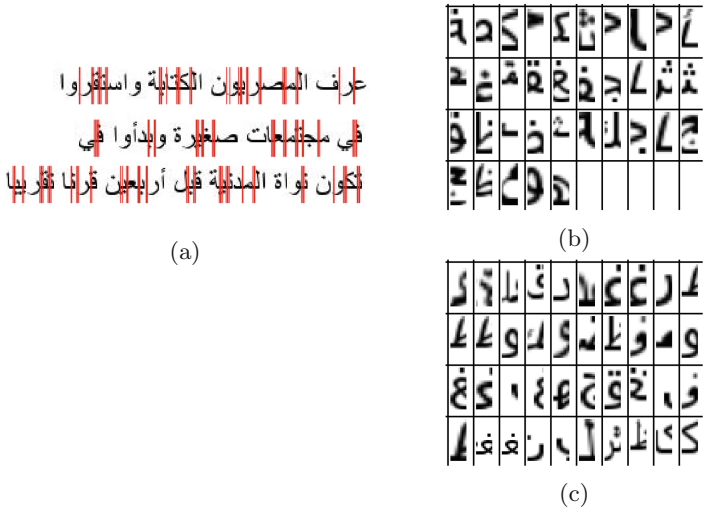
Table 2. Resulting values for loss functions and (Acc $L > 0$) accuracy of a test data with labels greater than 0.

	Recall	Validation	1-Font test		4-Font test	
	Loss	Loss	Loss	Acc L > 0	Loss	Acc L > 0
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 %

test sets in the table. The 1-Font model is able to segment 4-font test set with accuracy of 65.5 %, so it is able to get around 40 % of the other fonts' segments right.

The 3-Window model accuracy and experimental results from other segmentation models [9], are listed in Table 3. While the datasets used in all other experiment were not open and listed for reference, we compare our model directly to a 1-window mode on the same data, it's trained on one font data, by testing both on same 1-font test data. The 1-window model has 90.2 % accuracy which shows the huge significance of a 3-window model and its versatility against confusing windows that cause under and over segmentation.

Figure 7a 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. 7b), and the right channel in (Fig. 7c). The windows that maximally activate the left channel are mostly right



(a) (b) (c)

Fig. 7. (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).

Table 3. Comparison results of different methods

	Method	Data	Accuracy
Zidouri, Abdelmalek [23]	Structural method	200 images	90 %
Broumandnia et al. [24]	Wavelet transform	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, BMF and Spann, M [25]	Contour information	1,065 characters from each font are tested	97.01 %
Hamid, Alaa and Haraty, Ramzi [26]	Structural features for Feed-forward Multilayer neural networks	10,000 exemplars	69.72 %
Touj et al. [15]	HMM with standard Hough transform as method 1 and with generalized Hough transform as method 2	6,400 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	250,000 words	90.2 %
1-Font 3 windows model	**Our proposed multi-channel neural network**	**250,000 words**	**98.9 %**

parts of characters that appear next to a cut. The right channel learns left parts of the character that appear right to a cut place.

APTI Dataset. The APTI dataset subset of size 18pt from Arabic transparent font was run in the pipeline described in Sect. 4.4. The result is reported in Table 4. The character recognition model reported 94.8 % accuracy (5.2 % error), which leaves room for more improvement on the character recognition model we intend to do in our future work.

Table 4. Character recognition error rate on size 18pt subset of APTI.

System	Error (%)
Proposed pipeline	5.2
IPSAR [27]	4.3
UPV-REC1 [27]	3.1
Siemens [28]	0.0
THOCR [28]	0.9

5 Conclusion

A multichannel neural network is used to segment Arabic documents. It incorporates a sliding window as a middle channel and another next and previous windows for more context. This model acts very well against over-segmentation and under-segmentation problems. A model trained to segment one Arabic font has an accuracy of 98.9 %. A model trained to segment four fonts has a 95.5 % accuracy. This model showed better accuracy than using a one-window model of the Neural Network. Then the segmentation model output is used to classify characters using a character recognition model. This proposed pipeline was tested against an APTI dataset subset showing 94.8 % accuracy. Future work would be around improving the character recognition process in the pipeline.

The pipeline is useful for automatically building an open vocabulary Arabic OCR system, as the generation and training process can be autonomous and has very few variables or features to be tuned.

References

1. Al-Khateeb, J.H., Khelifi, F., Jiang, J., Ipson, S.S.: 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, November 2009
2. Al-Badr, B., Haralick, R.M.: 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. IEEE (1995)
3. Khorsheed, M.S., Clocksin, W.F.: Multi-font Arabic word recognition using spectral features. In: 15th International Conference on Pattern Recognition, Proceedings, vol. 4, pp. 543–546. IEEE (2000)
4. Jelodar, M.S., Fadaeieslam, M.J., Mozayani, N., Fazeli, M.: A persian OCR system using morphological operators. In: WEC (2), pp. 137–140 (2005)
5. Märgner, V.: Sarat-a system for the recognition of Arabic printed text. In: Proceedings of the 11th IAPR International Conference on Pattern Recognition. Conference B: Pattern Recognition Methodology and Systems, vol. II, pp. 561–564. IEEE (1992)
6. Azmi, R., Kabir, E.: A new segmentation technique for omnifont Farsi text. Pattern Recognit. Lett. **22**(2), 97–104 (2001)

7. Khoury, I., Giménez, A., Juan, A., Andrés-Ferrer, J.: Window repositioning for printed Arabic recognition. Pattern Recognit. Lett. **51**, 86–93 (2015)
8. Ahmad, I., Mahmoud, S.A., Fink, G.A.: Open-vocabulary recognition of machine-printed Arabic text using hidden Markov models. Pattern Recognit. **51**, 97–111 (2016)
9. Alginahi, Y.M.: A survey on Arabic character segmentation. Int. J. Doc. Anal. Recognit. (IJDAR) **16**(2), 105–126 (2013)
10. Amin, A.: Off-line Arabic character recognition: the state of the art. Pattern Recognit. **31**(5), 517–530 (1998)
11. Zheng, L., Hassin, A.H., Tang, X.: A new algorithm for machine printed Arabic character segmentation. Pattern Recognit. Lett. **25**(15), 1723–1729 (2004)
12. Bushofa, B.M.F., Spann, M.: Segmentation and recognition of printed arabic characters using structural classification. Image Vis. Comput. **15**, 167–179 (1997)
13. Nawaz, S.N., Sarfraz, M., Zidouri, A., Al-Khatib, W.G.: An approach to offline arabic character recognition using neural networks. In: Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems, ICECS, vol. 3, pp. 1328–1331. IEEE (2003)
14. Gouda, A.M., Rashwan, M.A.: Segmentation of connected Arabic characters using hidden Markov models. In: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, CIMSA, pp. 115–119. IEEE (2004)
15. Touj, S., Amara, N.B., Amiri, H.: Two approaches for arabic script recognition-based segmentation using the Hough transform. In: ICDAR, pp. 654–658. IEEE (2007)
16. Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: High-performance neural networks for visual object classification. arXiv preprint arXiv:1102.0183 (2011)
17. Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., Zhuowen, T.: Deeply-supervised nets. arXiv preprint arXiv:1409.5185 (2014)
18. Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng A.Y.: Multimodal deep learning. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 689–696 (2011)
19. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
20. Srivastava, N.: Improving neural networks with dropout. Ph.D. thesis, University of Toronto (2013)
21. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: Null, p. 958. IEEE (2003)
22. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: International Conference on Artificial Intelligence and Statistics, pp. 315–323 (2011)
23. Zidouri, A.: On multiple typeface Arabic script recognition. Res. J. Appl. Sci. Eng. Technol. **2**(5), 428–435 (2010)
24. Broumandnia, A., Shanbehzadeh, J., Nourani, M.: Segmentation of printed Farsi/Arabic words. In: IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2007, pp. 761–766. IEEE (2007)
25. Bushofa, B.M.F., Spann, M.: Segmentation of Arabic characters using their contour information. In: Proceedings of the DSP 97 13th International Conference on Digital Signal Processing, vol. 2, pp. 683–686. IEEE (1997)
26. Hamid, A., Haraty, R.: A neuro-heuristic approach for segmenting handwritten arabic text. In: ACS/IEEE International Conference on Computer Systems and Applications, pp. 110–113. IEEE (2001)

27. Al-Muhtaseb, H.A., Mahmoud, S.A., Qahwaji, R.S.: Recognition of off-line printed Arabic text using hidden Markov models. Signal Process. **88**(12), 2902–2912 (2008)
28. Slimane, F., Kanoun, S., El Abed, H., Alimi, A.M., Ingold, R., Hennebert, J.: ICDAR competition on multi-font and multi-size digitally represented Arabic text. In: 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 1433–1437. IEEE (2013)