Comparative Analysis of Raw Images and Meta Feature based Urdu OCR using CNN and LSTM

Asma Naseer, Kashif Zafar
Computer Science Department
National University of Computer and Emerging Sciences
Lahore, Pakistan

***Abstract*—Urdu language uses cursive script which results in connected characters constituting ligatures. For identifying characters within ligatures of different scales (font sizes), Convolution Neural Network (CNN) and Long Short Term Memory (LSTM) Network are used. Both network models are trained on formerly extracted ligature thickness graphs, from which models extract Meta features. These thickness graphs provide consistent information across different font sizes. LSTM and CNN are also trained on raw images to compare performance on both forms of inputs. For this research, two corpora, i.e. Urdu Printed Text Images (UPTI) and Centre for Language Engineering (CLE) Text Images are used. Overall performance of networks ranges between 90% and 99.8%. Average accuracy on Meta features is 98.08% while using raw images, 97.07% average accuracy is achieved.**

***Keywords*—Long Short Term Memory (LSTM); Convolution Neural Network (CNN); OCR; scale invariance; deep learning; ligature**

I. Introduction

The recognition of optical characters of cursive scripts always captures attention of computer scientists and linguists. Urdu is among those languages which use cursive script for writing. It is a widely spoken language which has 60,000,000 to 70,000,000 native speakers all around the world [3]. In Pakistan and in a few states of India, it is official language. Script of Urdu is Arabic, and Nastalique is the most popular font face. In this writing style, characters and ligatures are written at an angle of 45^o. Same character differs in shape due to certain reasons such as position of character within a ligature which can be start, middle or end. Adjacent characters also affect shape of same character. When same character joins a different character, it gets a different shape. In isolation, shape of a character is also different from its other shapes. Due to so many shapes of single character, there are so many unique shapes (characters and ligatures) in language. Characters are also overlapping which makes it hard to segment ligatures. Due to such characteristics of Urdu language and its script, Urdu OCR is still an open to research problem.

In this research, Urdu optical characters are recognized by using two different forms of inputs. The first form of input is raw images while the second form of input is graphs of ligature thickness. Two neural networks, LSTM and CNN, are trained on both these forms of input. Networks extract features from raw images and Meta features from formerly plotted graphs of ligature thickness. At the end, performance of networks using Meta features and raw images is compared and analysed.

LSTM shows considerably better results due to three factors, i.e. equation updating mechanism, memory cell and back propagation dynamics [4]. In this network, LSTM units are used instead of normal nodes. It contains memory cells consisting of self-feedback loops and three adaptive gates i.e. input, output and forget gate as can be seen in Fig. 1. Throughout training, forget gate takes decisions and eventually only important and relevant information is used for next iterations. Irrelevant or less important information is discarded [5], [6].

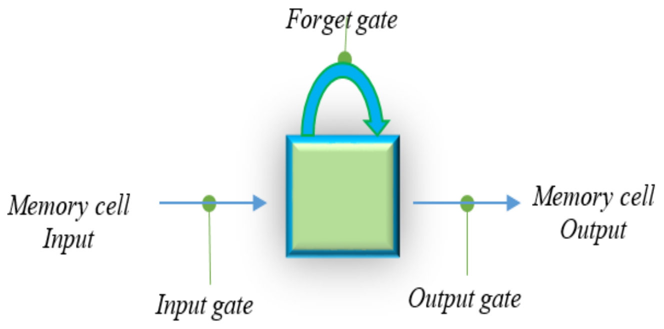


Fig. 1. LSTM, an illustration of memory cell.

CNN consists of different layers such as convolution layer, Rectified Linear Unit (ReLU) layer, SoftMax layer, Pooling layer, fully connected layer, input layer and output layer as can be seen in Fig. 2.These layers extract more relevant features from any type of input either multifaceted or simple. Convolution layers extract features in a progressive manner and pooling layers downscale feature space and memory. Fully connected layers connect two consecutive layers. As multiplication of small numbers arouses vanishing gradient problem so to resolve it ReLU layer is used. Other than these layers there are some more layers such as network in network layer, classification layer and dropout layer [6]-[8].

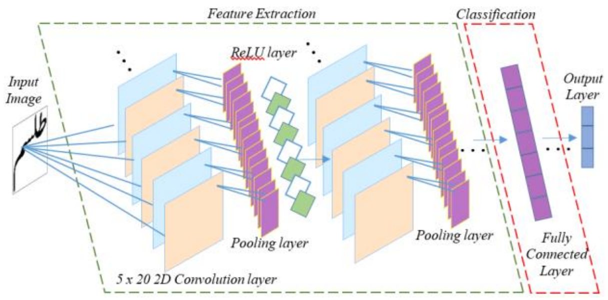


Fig. 2. An Illustration of CNN

Major contributions of this research are:

- Developing Meta Features
- Exploring scale invariance nature of ligature thickness graphs
- Analyzing and comparing performance of networks for Meta features and raw images.
- Comparing performance of CNN and LSTM
- Increasing accuracy of Urdu OCR up till 99.33%

Organization of paper is in such a way that, related work is described in Section II, Section III presents proposed methodology, results are reported in Section IV, Section V describes comparative analysis and finally conclusion and future work is given in Section VI.

II. Related Work

Although OCR is not a new field for research, and for certain languages like printed English, it is almost a solved problem, still for languages like Urdu; a lot of research is required. In early versions of Urdu OCR systems, Discrete Cosine Transformation (DCT) is calculated using overlapping windows [21]. These features are used to train Hidden Markov Model (HMM). As this system was trained only for a few classes of Urdu characters so it is further extended and all classes of Urdu characters are used [20]. In both these research work, artificially created data set of font size 36 is used. The system shows good performance on 36 font size but variation in font size affects accuracy. To handle font size variation, Tesseract is trained for different font sizes ranging from 12 to 36 [22]. Although performance of previous systems increased but for each font size an independent model is trained.

As there is a paradigm shift due to outstanding performance of deep learning algorithms, usage of deep networks in OCR becomes popular. Multilayer perceptron, CNN, Recurrent Neural Networks (RNN), LSTM and its variations are extensively used for OCR [9]-[13]. Raw images of UPTI [1] data set are used to train Multi-dimensional LSTM [13]. For the same data set, CNN is also trained and 98.1% accuracy is achieved. Text images are divided into different zones and 2DLSTM is trained to identify Urdu text lines [14]. This variation of LSTM is trained on density of pixels and it shows 93.39% accuracy. Other than Urdu, deep networks are widely used for other languages. Segmentation free data set, created synthetically for English and Greek, is used to train LSTM and 98.81% accuracy is achieved [15]. In a similar approach Bi-directional LSTM is trained for Fraktur (German) and English and 99.4% and 98.36% accuracy is achieved for English and German respectively [16]. LSTM shows 99% average accuracy for French, German and English [17]. This network is trained on normalized raw pixels values.

Besides deep networks, OCRopus, Tesseract and OCRoRACT are also very popular for different languages such as Latin, Urdu and Devanagri [9], [18], [19]. Segmentation based (when ligatures are divided into characters) and segmentation free (ligature based), both approaches are used for OCR systems for these languages. OCRopus, based on LSTM, is also trained on raw pixel values of Devanagri text images and 91% accuracy is achieved [11].

III. Proposed Methodology

For Urdu character recognition of different font sizes, two deep networks are used. Both networks are trained on raw images as well as on extracted feature i.e. ligature thickness graphs. CLE [2] and UPTI [1] text images are used for training and testing.

A. *Corpora*

Two Corpora, CLE Text Images and UPTI are used to train networks. From both corpora, in total 78,714 instances of text images are used. CLE corpus contains images from font size 14 to 40. For developing this corpus, 2912 text documents are scanned. In these text images, text font face is Nastalique. It is full of variety in different dimensions such as different eras, different paper quality, different printing, different ink quality and different domains etc. UPTI Corpus is a collection of 10 thousands text lines. From these lines, 771,339 characters are grouped in 44 classes.

B. *Meta Feature Extraction*

To recognize characters of different font sizes, thickness of ligatures and characters is measured. Trend of increase and decrease in thickness value remains same across different font sizes. When ligature thickness graphs are fed to networks they extract Meta features from them.

1) Extracting Thickness:

To extract thickness of ligatures and characters, certain steps are performed one after another. At first images containing text are converted into binary images with value only 1 (white) and 0 (black). From binary images, skeleton (centre line) of ligatures are obtained. At each point of skeleton, tangent line is attained by using current pixel and its next neighbouring pixel. For getting tangent line (1) and (2) are used. After that, perpendicular (normal line) at each tangent line is calculated using equation (3). This normal line is traversed and all back pixels on this line, in binary image, are counted. Number of pixels on this line are stored as thickness of ligature at that point.

$$g_i = \frac{\partial y}{\partial x} = \frac{(y_i - y_{i+1})}{(x_i - x_{i+1})} \qquad (1)$$

$$t_i = y_i - y_{i+1} = g_i\,(x_i - x_{i+1}) \qquad (2)$$

$$n_i = y_i - y_{i+1} = -\frac{1}{g_i}\,(x_i - x_{i+1}) \qquad (3)$$

Where, g_i is gradient, t_i is tangent and n_i is normal line at ith point.

For each image, a feature vector containing thickness is extracted. Variation in font size and ligature length results in feature vectors which are having different length. To make feature vectors of same size, normalization is performed. Vectors are either scaled up or scaled down while considering average length of all vectors. Once vectors of same length are obtained, graphs are plotted. To get graphs symmetrical for same character or ligature, smoothing is performed by using (4). Smoothing also decreases signal to noise ratio. For smoothing +/- 5 values are considered.

$$S_i = \begin{cases} \frac{1}{p\times2+1}\sum_{k=i-p}^{i+p} t_k & | p \le i \le s-p \\ \frac{1}{q\times2+1}\sum_{k=1}^{l} t_k & | i < p, 1 \le q \le p-1 \\ \frac{1}{q\times2+1}\sum_{k=q}^{s} t_k & | i > s-p, s-p+1 \le q \le s \end{cases} \quad (4)$$

Where, s is length of vector, t_i and S_i are thickness values at i^{th} point, before and after smoothing, respectively.

Finally, graphs of thickness are plotted and stored as images. The complete process of extracting thickness of ligatures and plotting them is given in Fig 3.

2) Raw Images

To explore effects of Meta features (ligature thickness graphs), raw images are also fed to networks. As images are of different sizes due to different font sizes and ligature length so all images are resized as per average height and width.

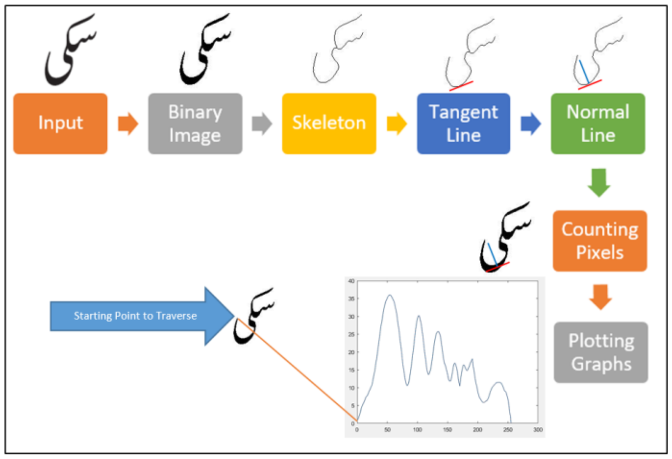


Fig. 3. Process of extracting thickness graphs.

C. Training

Two deep learning models, i.e. LSTM and CNN, are trained using thickness graphs and raw images so in total four models are trained. Eventually, performance of trained models is evaluated and compared. For training 80% data and for testing 20% data is used.

1) LSTM

At first, LSTM is trained on graphs of ligature thickness. Network is created with fully connected layers. Total number of fully connected layers are same as total number of classes. Input and output size are set as 16 and 100, respectively. Sigmoid is chosen as activation function of network. Network extracts Meta features from pre extracted features for training the model. Gating mechanism, mainly responsible for maintaining memory related issues, is handled using (5) to (9).

$$M_t^{\sim} = \tanh(W^M x_t + U^M hs_{t-1}) \quad (5)$$

$$Fg_t = \sigma(W^{Fg} x_t + U^{Fg} hs_{t-1}) \quad (6)$$

$$M_t = Fg_t \circ M_{t-1} + i_t \circ M_t^{\sim} \quad (7)$$

$$Og_t = \sigma(W_o x_t + U_o hs_{t-1}) \quad (8)$$

$$hs_t = Og_t \circ \tanh(M_t) \quad (9)$$

Where, $M_t^{\sim}$ is new memory cell, M_t is final memory cell, Fg_t is forget gate, Og_t is output gate and hs_t is hidden state.

Once LSTM is trained on thickness graphs, another model of same network is trained using raw images. Both models use same values for all properties and settings so that only thing that varies should be input.

2) CNN

CNN is also trained on both types of input, i.e. ligature thickness graphs and raw images. For both types of input, two independent models are trained while keeping all parameters same. For CNN network 2D convolution layers are created. Size of layer is set as 5×20. Other than convolution layer, input layer, ReLU layer, maximum pooling layers, softmax layer, classification layer and fully connected layers are also created. Size of batch is set as per total number of classes. Stochastic gradient descent with momentum is used. For convolution operation (10) is used.

$$op[i,j] = (w \times g)[i,j] = \sum_{p=-P}^{P}\sum_{q=-Q}^{Q} w[p,q]\, g[i+p, j+q] \quad (10)$$

Where, w is weight and g is input image.

IV. Results

After training networks, they are tested with 20% of data. Results of CNN and LSTM are described in Table I through Table IV. Total 16 results are reported for each model, considering length of ligature, corpus and ligature and character level accuracy.

Average performance of CNN for thickness input graphs, ranges from 94.91% to 99.22%. On the other hand, average performance of CNN for raw images is between 91.14% and 97.49%. Detail of results for this model, as per character and ligature length, can be seen in Tables I and II..

Table I describes accuracies, achieved by CNN for thickness graphs. CNN reveals same results for character recognition and ligature recognition for UPTI dataset but for CLE dataset, character level accuracy is better than ligature level accuracy by 2.27%. Results, shown by this model for raw images, also have the same trends. Accuracy for character recognition and ligature recognition is almost the same for UPTI dataset. For CLE dataset, character level accuracy is better than ligature level by 2.27%.

TABLE. I. RESULTS OF CNN FOR THICKNESS GRAPHS INPUT

Convolution Neural Network Input: Thickness Graphs				
Ligature Length	**UPTI**		**CLE**	
	Character Accuracy	**Ligature Accuracy**	**Character Accuracy**	**Ligature Accuracy**
1 Character	96.88%	96.88%	**100%**	**100%**
2 Characters	**100%**	**100%**	98.78%	98.44%
3 Characters	**100%**	**100%**	92.94%	88.71%
4 Characters	**100%**	**100%**	97.00%	92.50%
Average	**99.22%**	**99.22%**	**97.18%**	**94.91%**

TABLE. II. RESULTS OF CNN FOR RAW IMAGES INPUT

Convolution Neural Network Input: Raw Images				
Ligature Length	**UPTI**		**CLE**	
	Character Accuracy	**Ligature Accuracy**	**Character Accuracy**	**Ligature Accuracy**
1 Character	90.63%	90.63%	**100%**	**100%**
2 Characters	**100%**	**100%**	97.04%	96.24%
3 Characters	**100%**	**100%**	95.87%	94.64%
4 Characters	99.32%	99.20%	79.80%	73.68%
Average	**97.49%**	**97.46%**	**93.18%**	**91.14%**

TABLE. III. RESULTS OF LSTM FOR THICKNESS GRAPHS INPUT

Long Short Term Memory Network Input: Thickness Graphs				
Ligature Length	**UPTI**		**CLE**	
	Character Accuracy	**Ligature Accuracy**	**Character Accuracy**	**Ligature Accuracy**
1 Character	**100%**	**100%**	**100%**	**100%**
2 Characters	**100%**	**100%**	99.35%	98.76%
3 Characters	**100%**	**100%**	99.28%	98.76%
4 Characters	**100%**	**100%**	93.10%	87.25%
Average	**100.00%**	**100.00%**	**97.93%**	**96.19%**

TABLE. IV. RESULTS OF LSTM FOR RAW IMAGES INPUT

Long Short Term Memory Network Input: Raw Images				
Ligature Length	**UPTI**		**CLE**	
	Character Accuracy	**Ligature Accuracy**	**Character Accuracy**	**Ligature Accuracy**
1 Character	**100%**	**100%**	**100%**	**100%**
2 Characters	**100%**	99.20%	99.23%	99.17%
3 Characters	**100%**	**100%**	99.63%	96.75%
4 Characters	**100%**	**100%**	98.64%	96.71%
Average	**100.00%**	**99.80%**	**99.38%**	**98.16%**

Average performance of LSTM for thickness graphs is 100% for UPTI data set. For CLE data set average performance on same input is 96.19% for ligatures and 97.93% for characters. For raw images, LSTM reveals 98.16% to 100% accuracy as can be seen in Tables III and IV.

Table III presents accuracy shown by LSTM for thickness graphs. The model shows similar results for character and ligature level recognition for UPTI dataset. For CLE dataset, character level accuracy is 1.74% better than ligature level accuracy. In Table IV, performance of LSTM on raw images is described. LSTM reveals better accuracy for character level recognition by 0.2% for UPTI dataset, while for CLE dataset character level recognition is better than ligature level recognition by 1.22%.

For thickness graphs input, CNN shows 100% accuracy for 8/16 results while for raw images 6/16 results are 100% accurate. LSTM reveals 100% accuracy for all the eight results of UPTI corpus when thickness graphs are used. In total $\frac{10}{16}$ results are 100% accurate. For raw images LSTM provides $\frac{9}{16}$ 100% results.

Overall average performance of both networks is 98.08% for thickness graphs and 97.07% for raw images. Average performance of networks for raw images and thickness graphs for both corpora can be visualized in Fig. 4.

V. COMPARATIVE ANALYSIS

During this research two networks are trained on different forms of inputs from same data set so that effects of thickness graphs (Meta features) on deep networks can be analyzed. CNN reveals 2.82% better results for thickness graphs as compare to raw images. On the other hand LSTM reveals better performance for raw images by 0.80%. In Fig. 4, comparison of thickness graphs and raw images is given. In six out of eight experiments in which ligatures from length 1 to 4 are tested from both the corpora, accuracy of thickness graphs is better than raw images.

Results also reveal that performance of all the models for UPTI dataset is better in a consistent way. The reason is, UPTI contains cleaner (less noise) corpus as compare to CLE. CLE image dataset is also full of variations in different dimensions which results in a bit lesser accuracy as compare to UPTI dataset.

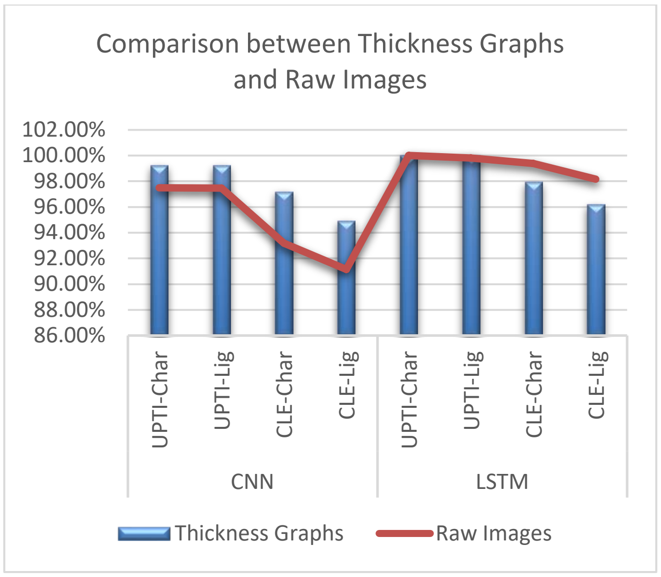
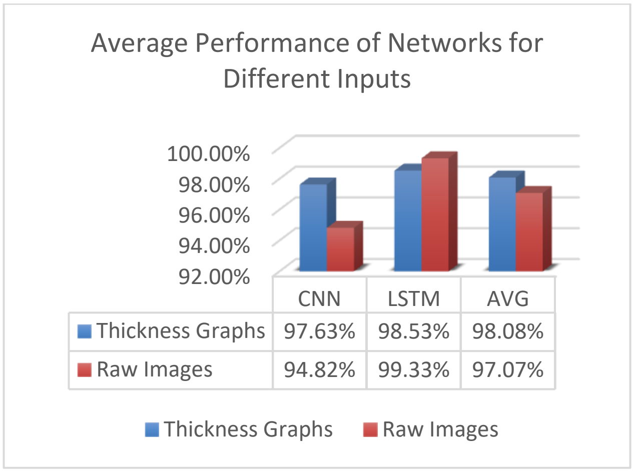


Fig. 4. Comparison between Thickness Graphs and Raw Images

Average performance of both networks is 97.07% for raw images and 98.08% for thickness graphs as can be seen in Fig. 5. Thickness graphs got better performance by 1.01%.



	CNN	LSTM	AVG
🟦 Thickness Graphs	97.63%	98.53%	98.08%
🟥 Raw Images	94.82%	99.33%	97.07%

Fig. 5. Average performance of networks.

VI. Conclusion And Future Work

In this research thickness graphs are used to extract Meta features to train deep networks. Thickness graphs show trend of thickness for a particular character or ligature. In certain cases, thickness graphs of different characters and ligatures may show same trends as can be seen in Fig. 6. In this figure two graphs are plotted for thickness of character '*alif*' ا and '*bay*' ب. Both graphs show almost same trend of thickness as low high and then again low. With a little bit more smoothing they may appear more similar. Such scenarios may arouse error. To avoid it, more features can be extracted such as direction of next pixel. In both graphs, illustrated in Fig. 6, there is same trend of thickness but it's in different direction. For letter '*alif*' direction is vertical while for letter '*bay*' direction is horizontal.

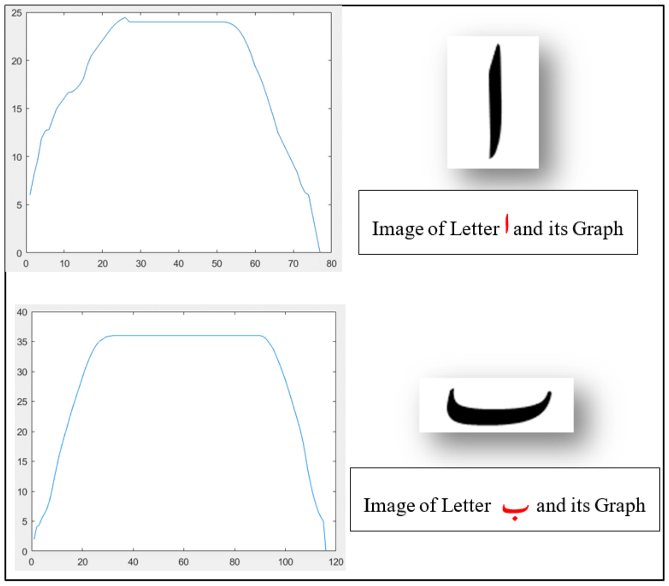


Fig. 6. Same trends in thickness graphs of letter ا and letter ب

Although overall average performance of networks shows that Meta features may bring more accuracy to deep learning frameworks, still more experiment can be carried out to explore its effects. In this research only one feature, thickness graphs, is used. This feature can be combined with more geometric or statistical features and results can be analysed.

References

[1] Sabbour, Nazly, and Faisal Shafait. "A segmentation-free approach to Arabic and Urdu OCR." In DRR, p. 86580N. 2013.

[2] Qurrat-ul-Ain, Niazi A., Farrah Adeeba, Urooj S., Sarmad Hussain and Shams S., "A Comprehensive Image Dataset of Urdu Nastalique Document Images", in the Proceedings of Conference on Language and Technology 2016 (CLT 16), Lahore, Pakistan

[3] Rahman, Tariq. "From Hindi to Urdu: A social and political history." Orientalistische Literaturzeitung 110, no. 6 (2015).

[4] Senior, Andrew W., and Anthony J. Robinson. "Forward-backward retraining of recurrent neural networks." In Advances in Neural Information Processing Systems, pp. 743-749. 1996.

[5] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term *memory." Neural computation 9, no. 8 (1997): 1735-1780.

[6] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.

[7] De Vries, Bert, and José Carlos Príncipe. "A theory for neural networks with time delays." In Advances in neural information processing systems, pp. 162-168. 1991.

[8] LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object recognition with gradient-based learning." Shape, contour and grouping in computer vision (1999): 823-823.

[9] Ul-Hasan, Adnan, Syed Saqib Bukhari, and Andreas Dengel. "OCRoRACT: A Sequence Learning OCR System Trained on Isolated Characters." In DAS, pp. 174-179. 2016.

[10] Ul-Hasan, Adnan. "Generic Text Recognition using Long Short-Term Memory Networks." (2016).

[11] Karayil, Tushar, Adnan Ul-Hasan, and Thomas M. Breuel. "A segmentation-free approach for printed Devanagari script recognition." In Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, pp. 946-950. IEEE, 2015.

[12] Ul-Hasan, Adnan, Syed Saqib Bukhari, Faisal Shafait, and Thomas M. Breuel. "OCR-Free Table of Contents Detection in Urdu Books." In Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on, pp. 404-408. IEEE, 2012.

[13] Naz, Saeeda, Arif I. Umar, Riaz Ahmad, Imran Siddiqi, Saad B. Ahmed, Muhammad I. Razzak, and Faisal Shafait. "Urdu Nastaliq recognition using convolutional–recursive deep learning." Neurocomputing 243 (2017): 80-87.

[14] Naz, Saeeda, Saad Bin Ahmed, Riaz Ahmad, and Muhammad Imran Razzak. "Zoning features and 2DLSTM for Urdu text-line recognition." Procedia Computer Science 96 (2016): 16-22.

[15] Ul-Hasan, Adnan, Muhammad Zeshan Afzal, Faisal Shafait, Marcus Liwicki, and Thomas M. Breuel. "A sequence learning approach for multiple script identification." In Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, pp. 1046-1050. IEEE, 2015.

[16] Breuel, Thomas M., Adnan Ul-Hasan, Mayce Ali Al-Azawi, and Faisal Shafait. "High-performance OCR for printed English and Fraktur using LSTM networks." In Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, pp. 683-687. IEEE, 2013.

[17] Ul-Hasan, Adnan, and Thomas M. Breuel. "Can we build language-independent OCR using LSTM networks." In Proceedings of the 4th International Workshop on Multilingual OCR, p. 9. ACM, 2013.

[18] Naz, Saeeda, Arif I. Umar, Riaz Ahmad, Saad B. Ahmed, Syed H. Shirazi, Imran Siddiqi, and Muhammad I. Razzak. "Offline cursive Urdu-Nastaliq script recognition using multidimensional recurrent neural networks." Neurocomputing177 (2016): 228-241.

[19] Ul-Hasan, Adnan, Saad Bin Ahmed, Faisal Rashid, Faisal Shafait, and Thomas M. Breuel. "Offline printed Urdu Nastaleeq script recognition with bidirectional LSTM networks." In Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, pp. 1061-1065. IEEE, 2013.

[20] A. Muaz, and S. Hussain, “Urdu Optical Character Recognition System”, MS Thesis, National University of Computer and Emerging Sciences, Lahore, Pakistan, 2010.

[21] S. T. Javed and S. Hussain, “Improving Nastalique Specific Pre-Recognition Process for Urdu OCR”, In Proceedings of 13th IEEE International Multitopic Conference (INMIC), Islamabad, Pakistan, 2009.

[22] Q. A. Akram, S. Hussain, A. Niazi, U. Anjum and F. Irfan, "Adapting Tesseract for Complex Scripts: An Example for Urdu Nastalique", In Proceedings of 11th IAPR Workshop on Document Analysis Systems (DAS 14), Tours, France, 2014.