Urdu Handwriting Recognition using Deep Leaning

Shehryar Malik*, M. Naeem Maqsood*, Abdur Rehman Ali*,
Ubaid Ullah Fayyaz* and Qurat-ul-Ain Akram†
*Department of Electrical Engineering,
University of Engineering and Technology, Lahore
†Center for Language Engineering, Lahore
shehryarmalik04@outlook.com, m.n.maqsood168@gmail.com, abdurrehmanali25@gmail.com,
ubaid@uet.edu.pk, ainie.akram@kics.edu.pk

***Abstract*—Optical character recognition aims to recognize text in images. Recent breakthroughs in deep learning have revolutionized OCR systems for languages such as English. However, their impact on Urdu has been minimal. This paper aims to bridge this gap. We develop a new dataset comprising of around $15,000$ images of Urdu handwritten text lines and use it to train different deep learning architectures. The first is the standard CNN-RNN architecture that optimizes the Connectionist Temporal Classification function. We also incorporate a trigram language model with this architecture to further improve performance. The second architecture is an attention-based encoder-decoder network that optimizes the cross-entropy function for each character in the transcription. We achieve accuracies of 91.51% and 90.07% on the two architectures respectively. These results are comparable to the state-of-the-art results on English datasets.**

***Index Terms*—Optical character recognition, deep learning, recurrent neural networks, connectionist temporal classification, attention-based model, language model**

I. Introduction

Consider the problem of reading text written on a piece of paper. The human mind accomplishes this task through a series of steps. It might begin by first identifying which parts of the paper contain text. Next, it would decide where to start reading from. To do that, it would also have to recognize the fact that the paper might contain many lines and that lines needs to be read individually in a certain sequential order. If a word is not written clearly then the mind knows to use previous and future contexts of the text to try to identify it. However, if the text is just a collection of random words, then the mind recognizes that and understands that it should not take into account the context. If the text is multilingual, then the mind also has to decide which parts of it belong to which language. It also needs to take into account cases where some of the languages are read left-to-right and the others right-to-left. In short, several complex decisions need to be made which also require an *understanding* of the text.

Recent breakthroughs in the field of machine learning, especially in the form of deep learning, have made systems that can read text documents more realizable than ever before. Apart from bringing artificial intelligence one step closer to human intelligence, these *optical character recognition* systems can assist people in a wide range of affairs. For example, state institutions can use these systems to digitize old records, thus sparing them the need of large storage areas. Similarly, libraries can easily create electronic versions of any old books they might have, thus preserving them for eternity.

Urdu is the national language of Pakistan and is spoken by over a 100 million people [1]. It is written from right to left using the Persian script and has 58 letters [2]. Characters physically join together to form ligatures. Each word contains one or more ligatures. The shape of each character varies according to its position in the ligature. Unlike English, no space is inserted between words in Urdu.

In this paper, we present an optical recognition system for Urdu using deep learning.

II. Literature Review

Optical Character Recognition (OCR) typically involves five steps: preprocessing, segmentation, feature extraction, classification and recognition and post-processing. In this chapter, we present a brief review of the OCR literature.

A. Preprocessing

Preprocessing involves a number of steps [3] that help improve the accuracy of later stages by removing noise and unnecessary details from an image.

Binarization is the process of converting a colored or gray-scale image to a binary image. In a binary image, all pixels can only take on two values: 0 or 1. One way of doing this is through Otsu's method [4] where we assume that an image contains two classes of pixels and then calculate the optimum threshold that separates them.

Sometimes scanning introduces a *skew* in images that needs to be corrected. Several different techniques exist for this purpose (see for e.g [5], [6]).

Other techniques involved in preprocessing include noise removal, background elimination, removal of black boundaries and extra white spaces, gray-scale normalization, size-normalization, smoothing and thinning (see for e.g. [3] and [7]).

B. Segmentation

Instead of feeding an image of an entire page of handwritten text to some classifier, it is usually useful to segment it into pieces first. These pieces could be individual lines, words or ligatures [3]. One way of doing this is through horizontal projection. Pixel values in each row are summed up. Assuming

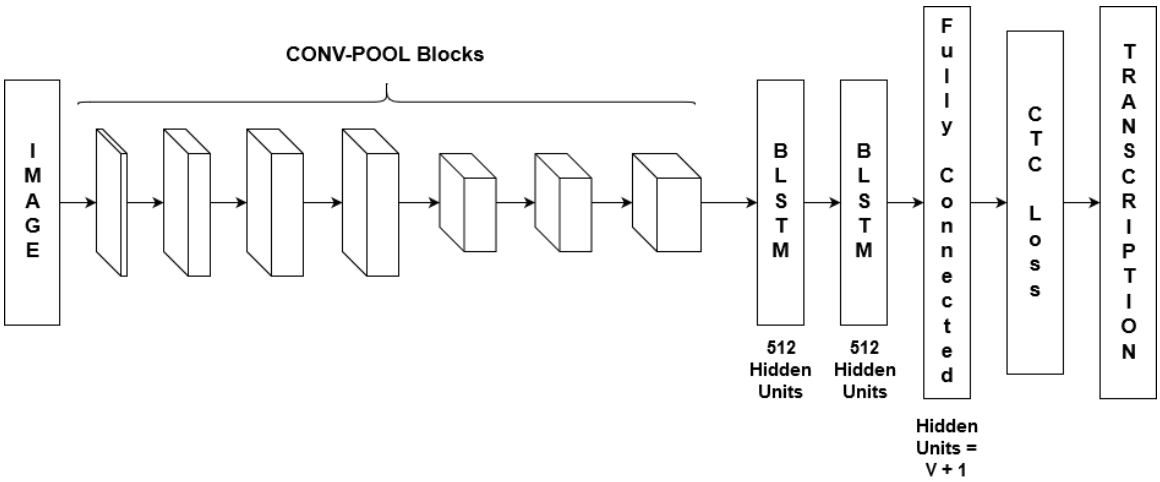


Fig. 1: The CNN-RNN-CTC Model

that 0 corresponds to a white pixel, a row summing up to zero would indicate a white line. This information can be used to segment the image into individual lines. Similarly, vertical projection can be used to segment images of lines into individual words and/or ligatures. However, in the case of images of handwritten text, this is generally harder (as lines/words/ligatures may overlap). [3] reviews different segmentation techniques.

C. Feature Extraction

Instead of feeding raw images (that might contain noise) to a classifier, one may first extract information that is relevant to the task-at-hand and only feed in that information. The goal of feature extraction is to extract this information from raw images.

Approaches to feature extraction include the computation of curvature, slope, end-points axes ratio, the length variations of strokes, shape context, discrete cosine transform and discrete wavelet transform and zoning features etc. (see for e.g. [8] and [3]).

However, more recent research does not extract features explicitly and instead feeds raw images to the classifier (see for e.g. [9] and [10]).

D. Recognition

Traditionally, Hidden Markov Models (HMM) were used for the recognition phase (see [11] for a review of some OCR systems that used HMM). However, recent research uses deep learning (neural nets) for this purpose.

Depending on the segmentation step, the classifier will either need to recognize images of either single characters ([12] and [13]) or complete lines ([5] and [14]) or entire paragraphs [9].

E. Post Processing

Once the text in an image has been recognized, additional steps such as spell-checking and grammar corrections can be carried out to improve the accuracy of the recognition system. This of course assumes that the text in the image is grammatically correct and contains valid words.

III. Dataset

[5] introduces the Urdu Nastaleeq Handwritten Dataset (UNHD). While the UNHD dataset consists of $10,000$ text lines, only $4,240$ are publicly available, which are not enough to train a robust deep neural network. As a result, we create a new dataset for the purpose of this thesis. The dataset will be available for further research.[1]

For this new dataset, $500,000$ text lines were selected from Urdu literature. $10,000$ lines were picked from these lines in such a way that the ratios of the frequencies of words remained the same. These lines (after some filtering) were divided into 490 pages, each consisting of 20 lines. Each page was given a unique 4-digit i.d. and was written by a distinct writer. Each writer too got a unique 4-digit i.d.

The writers ranged between 15 and 30 years of age, were of both sexes and mostly belonged to schools, colleges and universities. The writers were given pages with black lines drawn on them for writing. Red pens with 6 different stroke widths were used for writing. The writers were instructed to leave one blank line after every line. Writers usually took 1 to 3 lines to write each printed text line.

Each page was scanned using a flatbed scanner at 300 dots per inch (dpi) and saved using the $.jpg$ format. Only the red pixels were extracted from each page. This removed the black lines in the background. The images were then segmented into text lines using horizontal projection. Each image was assigned a unique 10 digit i.d. of the format $aaaa\_bbbb\_cc$, where $aaaa$ was the i.d. of the writer who wrote them, $bbbb$ was the i.d. of the 20-line page that the writer wrote and cc was the line number of the writer's page.

The final dataset contains $15,164$ text lines written by 490 different writers in 6 different strokes and has $13,497$ trigrams, $1,674$ bigrams and 61 unigrams.

[1]Contact http://cle.org.pk/ for this dataset.

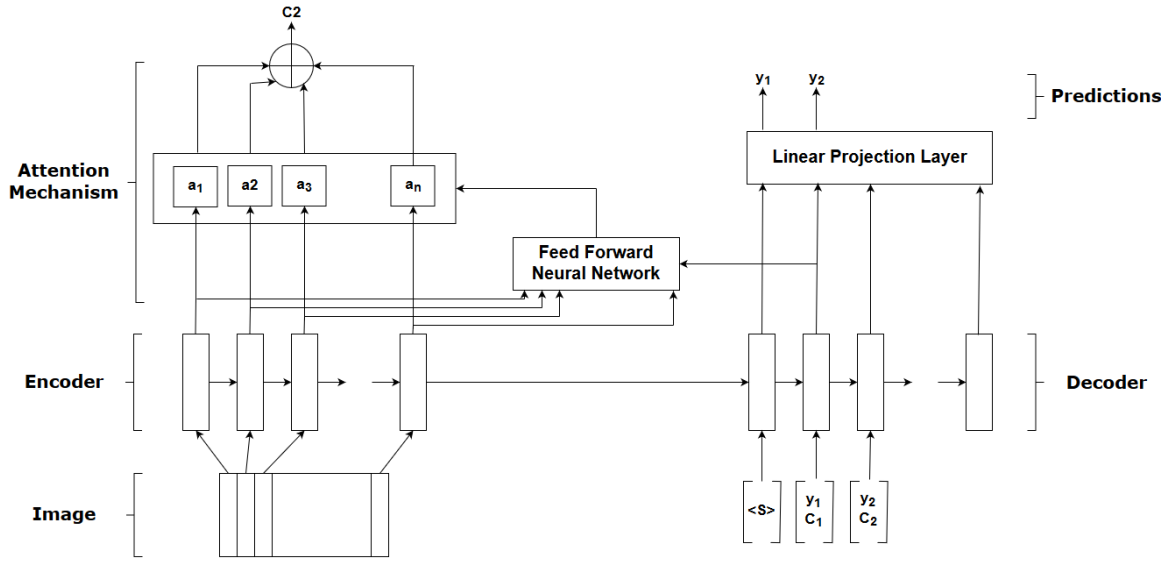


Fig. 2: The Attention-Based Encoder-Decoder Model

The dataset is further divided into training and test sets consisting of $13,351$ and $1,813$ images respectively. 440 writers contributed to the training set while 86 contributed to the test set. 288 images in the test set are of writers who also contributed to the training set.

IV. Experiments

All white columns in the images are removed. The images are then binarized using Otsu's method [4] and normalized to a height of 64. The width is adjusted in such a way that the aspect ratio is maintained. Images are divided into 5 buckets depending on their widths. All images in a bucket are zero-padded upto the maximum width in that bucket.

In Urdu, each character has a different shape based on its position (initial, in-between, final, isolated) in its ligature. We assign each *shape* a distinct i.d.

For all experiments, we use Adam [15] with an initial learning rate of 0.96. The batch size is set to 32. We use a dropout [16] of 0.2 for all recurrent neural networks. We also use gradient clipping [17]. The gradients are re-scaled whenever a gradient norm exceeds 5. No L2 regularization was used.

A. CNN-RNN-CTC Model

Fig. I shows the CNN-RNN-CTC model. As Urdu is read from right to left, we flip all images horizontally before feeding them to the network. Each CONV-POOL block contains a convolutional layer, a ReLU activation and a pooling layer in that order. In some cases, batch normalization [18] is applied to the output of the max pooling layer. Table I details the settings used for these blocks. The two numbers in the Pool Strides column correspond to the horizontal and vertical strides respectively. A stride of 1 essentially means that no pooling was done for that axis.

The learning rate was decayed by 0.96 after every $1,000$ training *steps* taken by the model. The connectionist temporal classification objective function [19] was optimized. We use three different decoding strategies: greedy search, beam search and beam search with language modeling. In the last two cases, the beam width is set to 10.

TABLE I

CONV-POOL Blocks in the CNN-RNN-CTC Model

Block	Filter Size	Pool Strides	Batch Norm
1	$5 \times 5 \times 32$	2,2	No
2	$5 \times 5 \times 64$	1,2	No
3	$5 \times 5 \times 128$	1,2	Yes
4	$5 \times 5 \times 128$	1,2	No
5	$3 \times 3 \times 256$	1,2	No
6	$3 \times 3 \times 256$	1,2	No
7	$3 \times 3 \times 512$	1,1	Yes

B. Language Model (LM)

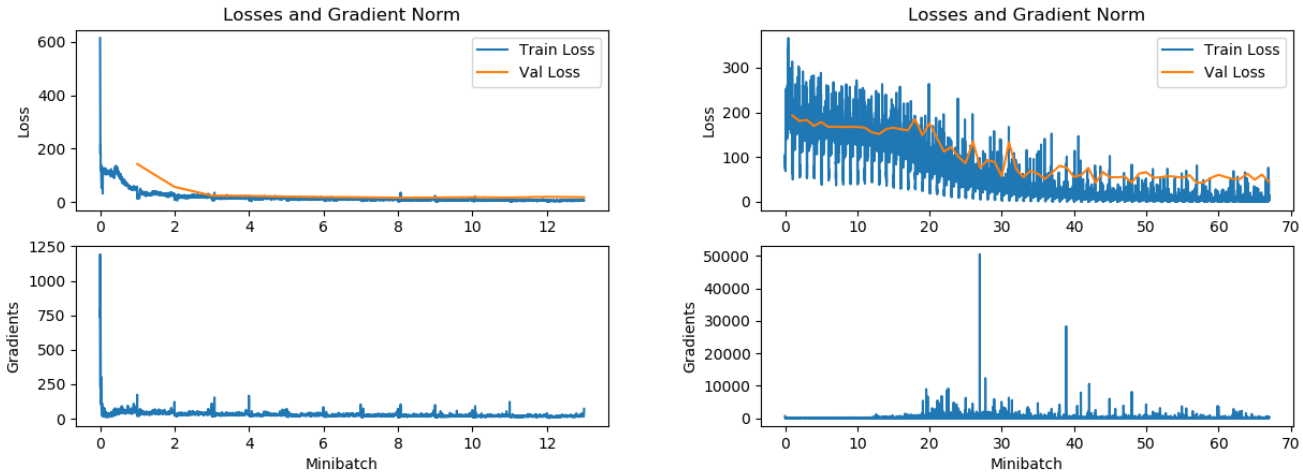
[20] presents an algorithm that incorporates a word-based language model for networks using connectionist temporal classification. However, word-based language models can only be used for languages such as English where words can be distinguished through the 'space' between them.

We instead use a ligature-based language model. Our language model is a simple trigram model with Kneser-Ney smoothing [21]. [20] essentially applies the language model whenever a space is encountered (indicating the end of a word). We instead apply the language model whenever a ligature ends. Recall that each character was assigned a distinct i.d. based on its position in its ligature. Therefore, we can easily identify all ids that signal the end of a ligature.

We set α and β in the algorithm in [20] to 0.5 and 4 respectively. Additionally, at each time step we only consider characters with probability greater than 0.001.

C. Attention-Based Encoder-Decoder Model

All images are fed to several CONV-POOL blocks. Table II details the settings used for these blocks. We feed the



(a) CNN-RNN-CTC

(b) Encoder-Decoder

Fig. 3: Training Plots

TABLE II

CONV-POOL Blocks in the Encoder-Decoder Model

Block	Filter Size	Pool Strides	Batch Norm
1	$5 \times 5 \times 16$	2,2	No
2	$5 \times 5 \times 32$	1,2	Yes
3	$5 \times 5 \times 64$	1,2	No
4	$3 \times 3 \times 64$	1,2	Yes
5	$3 \times 3 \times 128$	1,2	No
6	$3 \times 3 \times 128$	1,2	No
7	$2 \times 2 \times 128$	1,1	Yes

output of these blocks to an encoder-decoder network [22]. We also make use of the attention mechanism given in [23]. The encoder is two-layer stacked bidirectional LSTM where as the decoder is a two-layer stacked unidirectional LSTM. Each LSTM layer has 512 units. We use layer normalization [24] for both the encoder and the decoder. The alignment model is a simple feed forward network with 512 hidden units. We also assign each character id an embedding vector of size 256. It is this embedding vector that is fed to the decoder. We optimize the cross-entropy objective function for each character in the transcription. We do not flip images in this case and leave it to the model to learn the direction of the script.

Fig. 2 shows the model. We have omitted the convolutional and embedding layers for conciseness.

We make use of scheduled sampling [25] to train the entire model. The initial sampling probability is set to 0.80. Both this probability and the learning rate are decayed by 0.96 after every $4,000$ training steps. We use two decoding strategies: greedy search and beam search.

V. Results

Fig. 3 shows the losses and gradient norms during training.

Table III shows the results of the experiments.

Fig. 4 shows the alignments assigned to an image overtime by the alignment model. Note that the model learns to read from right to left

TABLE III

Results

	CNN-RNN-CTC	Encoder-Decoder
Greedy Search	88.50	89.52
Beam Search	88.75	**90.07**
Beam Search + LM	**91.51**	-
On English [14]	93.80	91.90

We train the trigram language model on $10,000$ Urdu text lines and achieve a perplexity of 47.621 on a held-out (test) set. Fig. 5 shows an example where the language model increases the accuracy of the output.

VI. Conclusion

In this paper, we have developed a dataset comprising of images Urdu handwritten text lines and their corresponding transcriptions and have used it to train to deep learning models. Additionally, we have also incorporated an n-gram language model.

References

[1] "Urdu," http://www.omniglot.com/writing/urdu.htm, accessed: 2019-04-08.

[2] "Controversy over number of letters in Urdu alphabet," https://www.dawn.com/news/919270, accessed: 2019-04-08.

[3] S. Naz, K. Hayat, M. I. Razzak, M. W. Anwar, S. A. Madani, and S. U. Khan, "The optical character recognition of urdu-like cursive scripts," *Pattern Recogn.*, vol. 47, no. 3, pp. 1229–1248, Mar. 2014.

[4] N. Otsu, "A Threshold Selection Method from Gray-level Histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[5] S. B. Ahmed, S. Naz, S. Swati, and M. I. Razzak, "Handwritten urdu character recognition using one-dimensional blstm classifier," *Neural Computing and Applications*, pp. 1–9, 2017.

[6] A. Dengel and R. Ahmad, "A novel skew detection and correction approach for scanned documents," 04 2016.

[7] Y. Alginahi, "Preprocessing techniques in character recognition," in *Character Recognition*, M. Mori, Ed. Rijeka: IntechOpen, 2010, ch. 1.

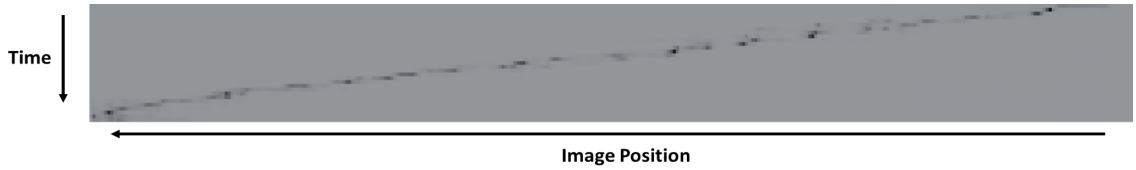


Fig. 4: Visualization of the Attention Mechanism

(a) Image

نے ان کی کسی شادع پہ کوئی گھونسلل بنایا۔

(b) Without a Language Model

نے ان کی کسی شاخ پہ کوئی گھونسلہ بنایا۔

(c) With a Language Model

Fig. 5: Demonstrating LM

[8] A. Lawgali, Bouridane, M. Angelova, and Z. Ghassemlooy, "Handwritten arabic character recognition: Which feature extraction method," *International Journal of Advenced Science and Technology*, vol. 34, pp. 1–8, 01 2011.

[9] T. Bluche, J. Louradour, and R. Messina, "Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention," 11 2017, pp. 1050–1055.

[10] M. Jain, M. Mathew, and C. V. Jawahar, "Unconstrained ocr for urdu using deep cnn-rnn hybrid networks," *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 747–752, 2017.

[11] L. M. Lorigo and V. Govindaraju, "Offline arabic handwriting recognition: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 712–724, May 2006.

[12] A. Elsawy, M. Loey, and H. El-Bakry, "Arabic handwritten characters recognition using convolutional neural network," *WSEAS TRANSACTIONS on COMPUTER RESEARCH*, vol. 5, pp. 11–19, 01 2017.

[13] A. Sahlol and C. Suen, "A novel method for the recognition of isolated handwritten arabic characters," *CoRR*, vol. abs/1402.6650, 2014.

[14] A. Chowdhury and L. Vig, "An efficient end-to-end neural model for handwritten text recognition," in *BMVC*, 2018.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[16] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. III–1310–III–1318.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 369–376.

[20] A. L. Maas, A. Y. Hannun, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *CoRR*, vol. abs/1408.2873, 2014.

[21] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling." in *ICASSP*. IEEE Computer Society, 1995, pp. 181–184.

[22] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv*, 2014.

[24] L. J. Ba, R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.

[25] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 1171–1179.