P-ISSN: 2618-0723
E-ISSN: 2618-0731

NAAS Rating: 5.04
www.extensionjournal.com

International Journal of Agriculture Extension and Social Development

Volume 7; Issue 11; November 2024; Page No. 101-107

Received: 01-09-2024
Accepted: 12-10-2024

Indexed Journal
Peer Reviewed Journal

Handwritten word recognition with deep learning

Sonali Das, Subhrajit Ray, Sourajit Acharya, Prasanta Kumar Barik

Department of Mechanical and Electrical Engineering, CAET, OUAT, Bhubaneswar, Odisha, India

DOI: https://doi.org/10.33545/26180723.2024.v7.i11b.1296

Corresponding Author: Subhrajit Ray

Abstract

Deep learning, the study of multi-layer artificial neural networks, had a significant impact on a number of artificial intelligence domains, including handwriting recognition. In this work, several insights from the domains of deep learning and handwriting recognition were focused. This paper seeks to classify an individual handwritten word so that handwritten text can be translated to a digital form. To complete this objective, two basic strategies were employed: firstly, character segmentation and secondly, classifying the words. For the former, to train a model that can correctly categories words, Recurrent Neural Network (RNN) was used with various architectures. For the latter, bounding boxes were created for each character using Long Short Term Memory Networks (LSTM) with convolution. After segmenting the characters, data were fed to a Convolution Neural Network (CNN) for classification. The IAM handwriting dataset was primary source of training the data for handwriting recognition system. Over 96,000 samples of handwritten text are included in this dataset. Based on the classification and segmentation outcomes, each word was reconstructed.

Keywords: RNN; Long Short-Term Memory networks (LSTM); Handwriting recognition

Introduction

Machine learning technique is the method of gaining knowledge from data by identifying key characteristics or trends to help in decision-making. It has influenced many facets of society, from search-engine recommendations for e-commerce websites to a wide range of smart device applications including voice to text transcriptions, facial recognition, and speech recognition [1]. The method employed to address these challenging issues is known as deep learning.

Algorithms that segment the word images in a given line image can be integrated with this study, and these algorithms can then be used with algorithms that segment the line images in a given image of an entire handwritten page. By include these, our study may provide a deliverable that an end user could use. This deliverable would be a fully working model that would ask the user to snap a picture of a page of notes in order to assist them in solving the difficulty of translating handwritten documents into digital format. It is important to note that, although our model requires additional layers to produce a fully functional deliverable for an end user, it was believed that the classification portion of the problem is the most challenging, which is why it was chosen to focus on it rather than segmenting lines into words, documents into lines, etc. This problem was tackled using full word pictures, as RNN perform better on unprocessed input pixels. It was attempted to build upon our findings by extracting characters from each word image and then categorising each character separately in order to recreate a whole word. To put it briefly, these models get an image of a word and produce its digital representation.

Existing Systems

Digit categorization for postal mail was the primary motivation behind the classification of handwritten text. In order to identify monospaced fonts, Jacob Rabinow's early postal readers used scanning technology and hardwired circuitry. This was enhanced and an advanced scanner was created that allowed for greater flexibility in text composition and encoded the data onto a barcode that was printed on the letter [2].

OCR software could recognise any typeface. Using this program, a more advanced use of the matrix technique (pattern matching) was made [3]. In essence, this would compare the read character's bitmaps with the bitmaps of the template character in order to identify which character it most closely matches. The software's sensitivity to size differences and the unique characteristics of each writer's style was a drawback. OCR software started employing feature extraction instead of templating to enhance the templating. Software would search for characteristics like geometric moments, zoning, and projection histograms for each letter [4].

Some of the first end-to-end contemporary deep learning models were inspired by the work of Lecun *et al*., who concentrated on applying gradient-based learning approaches employing multi-module machine learning models [5]. The next significant advancement in achieving high OCR accuracy was the OCR task's usage of a Hidden Markov Model. By using letters as a state in this method, the context of the character can be taken into consideration while figuring out the next hidden variable [6]. In comparison to feature extraction methods and the Naive Bayes methodology, this resulted in increased accuracy [7].

Data Collection

The International Conference on Document Analysis and Recognition (ICDAR) provided the IAM dataset that was used in this experiment. It is an international academic conference which is held every two years in a different city. It deals with the recognition of characters and symbols, text in print or handwriting, graphics analysis and recognition, document analysis, document comprehension, digital libraries and historical documents, document-based forensics, and text analysis of scenes captured on film and camera. The IAM Handwriting Database includes handwritten English text formats that can be utilized for writer identification and verification experiments, as well as for training and testing handwritten text recognizers [8].

The database was initially made available at the 1999 ICDAR. Using this database an HMM based recognition system for handwritten sentences was developed and published at the ICPR 2000 [9].

Unrestrained handwritten text forms were scanned at 300 dpi and saved as PNG images with 256 grey levels, which are included in the database. This dataset includes handwritten text from more than 1500 forms, each of which is a piece of paper containing lines of text contributed to by more than 600 writers, totalling more than 11500 words and 5500 sentences. After that, the words were segmented and manually checked, the corresponding XML files contain all of the form label metadata.

The original text was derived from the Lancaster-Oslo/Bergen (LOB) corpus, which has over a million words of complete English phrases. There are 1,066 forms in the database as well, written by about 400 different authors. IAM Handwriting Dataset was chosen as the source the training, validation, and test data for the models because of this database's quality, depth, and breadth, which make it a good choice for many handwriting recognition applications.

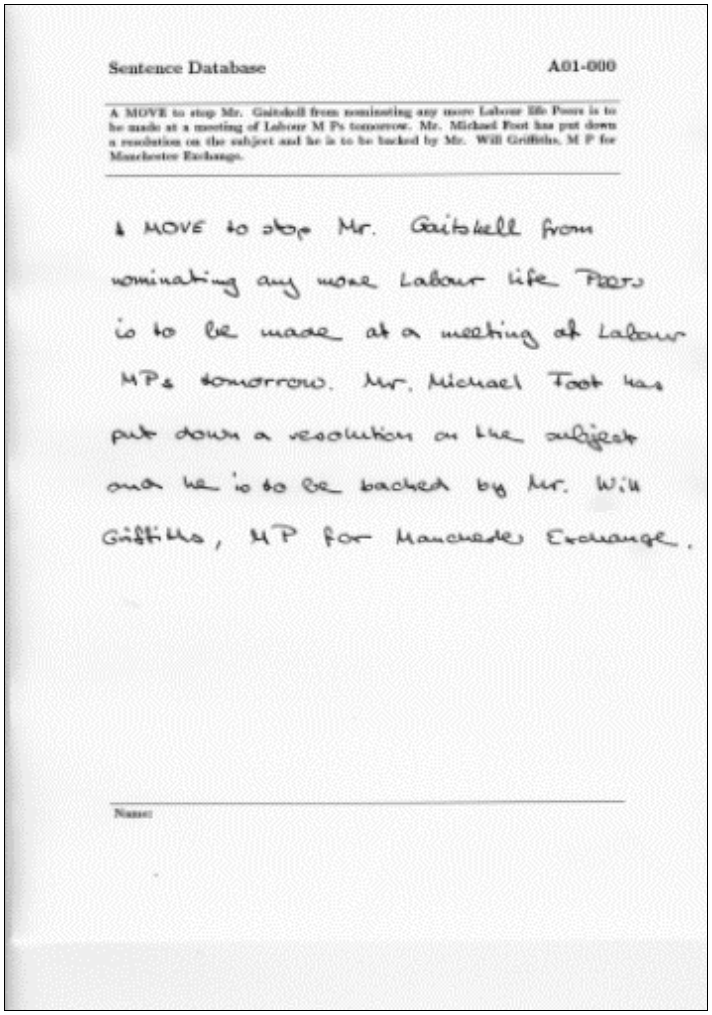

Fig 1: an example form from the IAM Handwriting dataset

Proposed Methods

Preprocessing

The initial stage of the recognition system might be regarded as the preprocessing phase. Rearranging the photos in a way that facilitates the recognizer's learning process will be the primary objective of this step. For every word image, the subsequent preprocessing procedures was used.

a. Image Augmentation
b. Padding images
c. Image down sampling

To provide the model with a greater variety of training samples, image augmentation involves manipulating photographs to create distinct versions of content. Image augmentation manipulations are forms of image preprocessing, but there is a crucial distinction between

image preprocessing and image augmentation manipulations. Image preprocessing processes are applied to both training and test sets, whether image augmentation is limited to the training set. Thus, a transformation that could be an augmentation in some situations may best be a preprocessing step in others.

The decision was made to crop the whitespace-filled photos to the largest possible width and height inside the dataset. In the process, equal amounts of white space were added to the width and height measurements. Although the images of the same words would essentially be padded with similar sizes, this method does not always alter the intrinsic characteristics of the word images, nor does it alter the relative relationship between the images.

In image analysis, compression transmission is frequently utilised. It maintains the two-dimensional image representation while lowering the spatial resolution of the images [10]. Usually, it is used to minimize computation time and storage. There are many ways for down sampling an image such as resize and downscale. Image downscale uses a factor of n x n to rescale the image. The mean of these components is used to replace each block with its mean.

Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs), a specific kind of neural network distinguished by the existence of self-connected nodes, form the foundation of the methodology offered in this study. These nodes give the network the ability to remember and monitor past inputs, which enables it to store and retrieve data for extended periods of time. Because word characters cannot be thought of as truly independent components, this specific feature enables the RNN to learn the context of the labels, which proves to be a huge advantage in handwriting recognition. Images containing handwritten text are processed as time sequences along single or multiple axis in order to apply RNNs for handwriting recognition.

Since the vanishing gradient problem prevents classical RNNs from storing large amounts of data, their capacity to retain historical content is still severely constrained. [11]. It is actually difficult for an RNN to close gaps of more than around 10 time steps between the relative goal events and the relevant input's presentation. Long Short-Term Memory (LSTM) was employed in order to solve this issue [12]. LSTMs are RNN nodes that are specifically designed to maintain long-term dependency. They are made up of three gates that regulate the node's input and output as well as a self-connected memory cell that resembles a traditional RNN node. The input to the LSTM node is actually a sigmoid function for each gate. The node's ability to receive fresh input is determined by the first gate, called input gate. Activation values of the memory cell can be reset by the node through the use of the second gate, a forget gate. The final gate governs which portions of the cell output are accessible to the subsequent nodes. It is an output gate.

By using opposite two-directional layers, or so-called Bidirectional Long Short-Term Memory (BLSTM), RNN models based on LSTM can be further improved. The forward layer processes the sequence from beginning to end with the aim of learning the input's context, whereas the backwards layer does the opposite by processing the sequence from end to beginning. It was demonstrated [13] that this architecture performs better than a simple uni-directional LSTM. The probabilities of the classes were scored using the *softmax* activation function. The output layer's categorical cross entropy was employed to reduce error (loss) during training.

The global minimum was determined using the Adam optimisation tool. There were only 80 epochs, therefore the learning rate was set to 0.001. The learning rate should be lower if there are more eras. During training, accuracy and loss were employed as the metrics to be assessed in each epoch.

Connectionist Temporal Classification approach

Since both architectures demand a direct alignment between the input characteristics and goal variables, the feed-forward technique and the original recurrent neural networks are similar in this regard. However, segmenting text into words rather than individual letters is far easier in handwriting recognition challenges seen in real life. A previous segmentation step would be necessary in order to achieve direct alignment between the input character picture and the character target label. Being an extremely difficult problem on its own, it becomes more complex with time due to the increased likelihood of potential errors occurring even during the segmentation process. Consequently, this would also restrict the context that the RNN could learn about the data.

The Connectionist Temporal Classification (CTC) method was developed to address this issue, it was first used for speech recognition [14] and then expanded to include handwriting recognition [15]. By interpreting the network's output as a probability distribution over all potential label sequences on the given input sequence, CTC allows one to avoid the previously mentioned direct alignment between the input variables and the target labels.

A *softmax* output with N +1, where N is the entire number of labels, makes up the network's last layer (in case of handwriting recognition - characters). These outputs define the probability of observing each label or not observing a label at a given moment in time. One can acquire a set of probabilities that correspond to a possible output sequence given an input sequence by using single character probabilities.

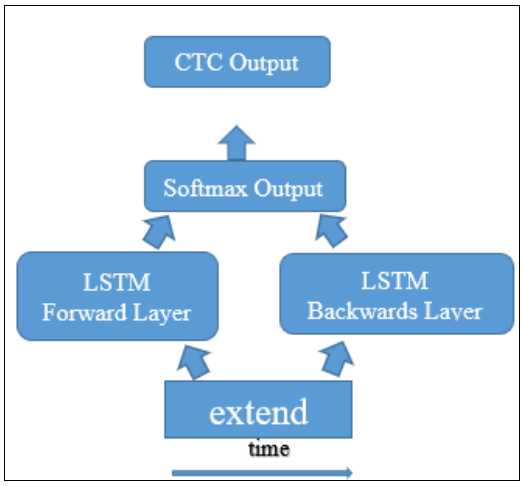

Fig 2: RNN architecture with Connectionist Temporal Classification output layer

Experiments

Building the character vocabulary

The *String lookup* layer was used for building the character vocabulary. With optional out-of-vocabulary management, this layer uses a table-based lookup to convert a set of arbitrary strings into an integer output.

Data Separation and Loading

The images from the IAM dataset were divided into three sets: 90% as Training set, 5% as Validation set and the remaining 5% as Test set. The training set was used to learn the features that distinguish the styles. The validation set was used to check the model accuracy during training, and the test set was used as unseen data by the model to test performance and accuracy of the model.

Model Training

Using the train dataset as input, the model was trained. The function accepts as a parameter the number of steps per epoch. Here the step is equal to the number of batches, which is 64. The accuracy and loss of the model on the training data from every epoch are the function's outputs. A deep learning model was created. Our model was based on the RNN. We made some essential modifications to the model to fit the IAM dataset.

The designed model is sequential that means all layers are added and trained sequentially. Every patch from the complete training set was used as the model's input. Data loading in batches was made possible by using the data generators. Batch size 64 was used. Thus, the input layer has a tensor of 64 patches of 128x32 pixels and one channel per batch in case of the IAM dataset.

A single average Pooling layer and a single Dropout layer followed each block of RNN layers, with the exception of the final block, which lacks a Dropout layer. Each block of RNN layers consists of two or three layers. In the first block, the number of kernels was set to 32, then it was doubled in each subsequent block. The RNN layers were employed as a feature detector. The *ReLU* activation function was used in all the RNN layers. This is because *ReLU* operates more quickly than other comparable activation functions.

Furthermore, dropout layers were introduced after each RNN block until the last one in order to avoid the model from overfitting for regularization purposes. 20% of the input units were eliminated from the model by setting the dropout rate to 0.20.

Results

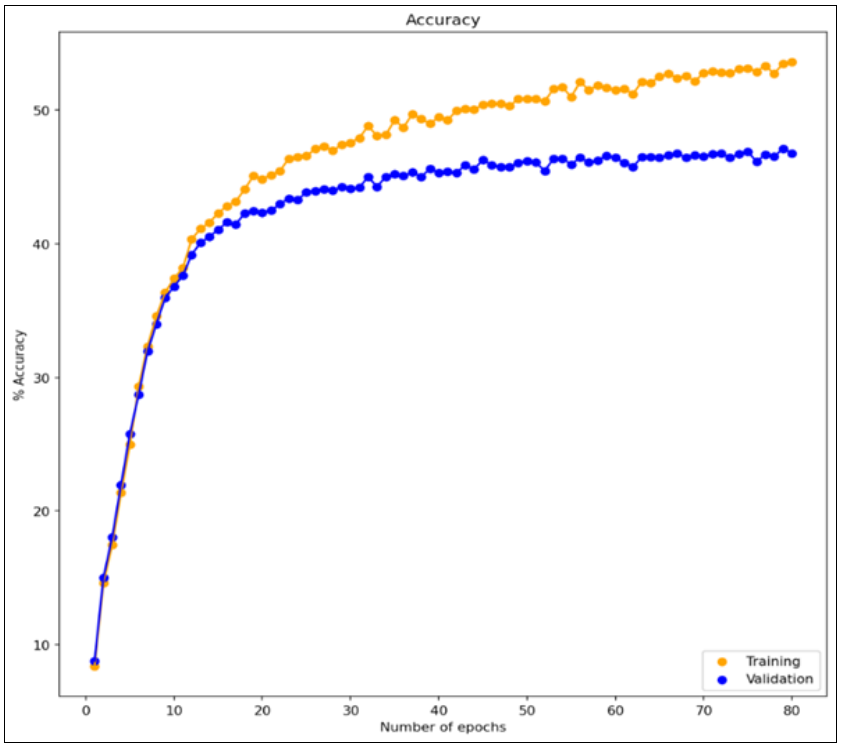

Fig 3: Accuracy and validation accuracy of model in 80 epochs

Accuracy Vs validation

The model was trained on the IAM dataset had an overall accuracy of 74% on the training set using 80 epochs. The model accuracy started 8% in the first epoch and increased slowly to 14% in the second epoch. In the next epochs it increased the accuracy more gradually until reached 74%. Interestingly, the validation accuracy also increased with the training accuracy (Figure 3). Similarly, the loss of the model decreased from 13.75 to 1.4. The model had accuracy of 75% on the validation set.

Impact of similarity between a pair of sequences on accuracy

To keep track of how similar two sequences are to one another, a function was employed. A metric called edit distance is used to determine how similar two sequences are to one another. In the initial era, it returns 20.49, then in the subsequent epoch, it gradually dropped to 20.13. The accuracy dropped more gradually during the ensuing epochs, reaching 17.29. Figure 23 illustrates the minimal decrease in the edit distance metric after 10 epochs.

Training edit distance is almost constant after 10 epochs. This indicates that word sequence edit distance disappears after 10 epochs. The performance of the model increases faster at the beginning and steady after 10 epochs

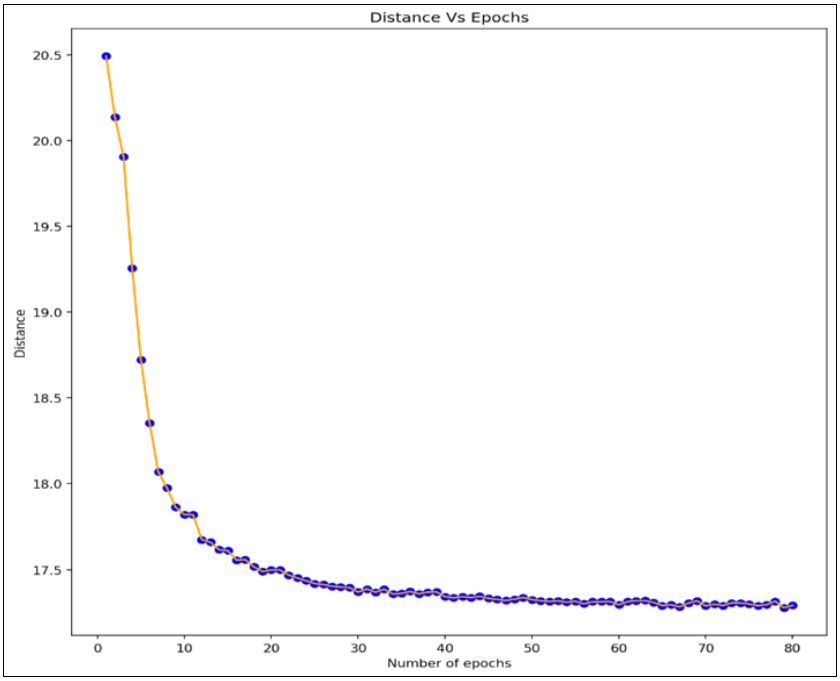

Fig 4: Edit Distance in 80 epochs

Table 1: f1-score for Training batch

Accuracy & f1-score for Training batch			
Sl No	**Accuracy in %**	**f1-score**	**f1-score in %**
0	82.813	0.822	82.188
1	84.375	0.833	83.267
2	84.375	0.837	83.705
3	89.063	0.890	88.977
4	76.563	0.754	75.417
5	92.188	0.929	92.909
6	84.375	0.844	84.375
7	82.813	0.828	82.813
8	89.063	0.884	88.393
9	81.250	0.813	81.250
10	90.625	0.906	90.625
11	84.375	0.839	83.854
12	89.063	0.891	89.063
13	85.938	0.859	85.938
14	87.500	0.875	87.500
15	89.063	0.891	89.063
16	82.813	0.821	82.143
17	87.500	0.870	86.979
18	84.375	0.849	84.896
19	78.125	0.781	78.099
20	89.063	0.889	88.862
21	89.063	0.885	88.542
22	82.813	0.823	82.292
23	81.250	0.811	81.101
24	92.188	0.916	91.563
25	87.500	0.875	87.500
26	82.813	0.828	82.813
27	84.375	0.844	84.375
28	84.375	0.838	83.750
29	76.563	0.764	76.414

Table 2: f1-score for Validation batch

Accuracy and f1-score for Validation batch		
Accuracy in %	**f1-score**	**f1-score in %**
62.500	0.625	62.500
82.813	0.828	82.813
76.563	0.766	76.563
68.750	0.688	68.750
76.563	0.766	76.563
76.563	0.766	76.563
73.438	0.737	73.698
76.563	0.766	76.563
78.125	0.775	77.455
84.375	0.844	84.420
65.625	0.661	66.146
75.000	0.750	75.000
81.250	0.801	80.060
78.125	0.784	78.438
75.000	0.755	75.521
81.250	0.806	80.580
64.063	0.649	64.861
73.438	0.734	73.438
68.750	0.688	68.750
70.313	0.690	69.018
78.125	0.781	78.080
73.438	0.740	73.958
78.125	0.782	78.229
87.500	0.875	87.500
75.000	0.744	74.375
76.563	0.749	74.858
76.563	0.766	76.563
79.688	0.797	79.688
79.688	0.797	79.688
68.750	0.682	68.229

Table 3: f1-score for Testing batch

Accuracy and f1-score for Testing batch		
Accuracy in %	**f1-score**	**f1-score in %**
78.125	0.768	76.786
70.313	0.709	70.938
78.125	0.786	78.646
76.563	0.759	75.938
78.125	0.768	76.830
73.438	0.734	73.438
71.875	0.724	72.396
73.438	0.734	73.438
67.188	0.677	67.684
78.125	0.781	78.125
78.125	0.780	77.951
71.875	0.737	73.735
70.313	0.698	69.792
70.313	0.697	69.688
71.875	0.719	71.875
67.188	0.671	67.083
79.688	0.813	81.250
65.625	0.648	64.826
71.875	0.714	71.373
68.750	0.693	69.271
73.438	0.740	73.999
67.188	0.701	70.052
76.563	0.753	75.313
75.000	0.750	75.000
70.313	0.696	69.643
81.250	0.813	81.250
75.000	0.750	75.000
65.625	0.663	66.319
85.938	0.860	86.042
73.438	0.733	73.289

Model Accuracy

F1-score, is a measure of a model's accuracy on a dataset. Using each of the 30 batches of data sets, we obtain the f1-score and accuracy for training, validation, and testing to validate the model.

Consistent with the model's overall accuracy, the model also demonstrated strong word recognition across the entire batch. The F1-score of the sets was between 75% and 92%. Similarly, the classifier had a good balance between sensitivity and specificity for all the batches. In the training batch f1-score was between 75% to 92% and accuracy was between 76% to 92%. In the testing batch f1-score was between 65% to 86% and accuracy was between 66% to 86%. In the validation batch f1-score was between 63% to 88% and accuracy was between 63% to 88%.

Conclusion

In this research, the model was proposed that automatically scans handwritten text and converts it to digital representation. The intricate patterns and features of old manuscripts and cursive writing are challenging to translate into digital media. However, it was found that the key to improving the model is to use image-processing techniques, particularly padding. Conclusion is that the optimum option for handwritten text recognition is to combine deep learning and image processing. Also, changing hyperparameters especially increasing number of epochs from 50 to 80 and increasing number of learning layers up to ten enabled the model to avoid overfitting and underfitting respectively. Applying the approach on an external dataset from the IAM showed the reliability of the model (avg. accuracy = 75.41%).

The results reveled that most words are recognized with high accuracy (avg. accuracy = 75%) except those images are not in a very bad condition.

References

1. Angra S, Ahuja S. Machine learning and its applications: A review. In: 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC); c2017 Mar; 57-60. IEEE.
2. Mahadevkar S, Patil S, Kotecha K. Enhancement of handwritten text recognition using AI-based hybrid approach. MethodsX. 2024;12:102654.
3. Plamondon R, Srihari SN. Online and off-line handwriting recognition: a comprehensive survey. IEEE Trans Pattern Anal Mach Intell. 2000;22(1):63-84.
4. Tiwari S. An introduction to QR code technology. In: 2016 International Conference on Information Technology (ICIT); c2016 Dec; 39-44. IEEE.
5. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436-444.
6. Graves A, Schmidhuber J. Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems. 2008;21.
7. Krevat E, Cuzzillo E. Improving off-line handwritten character recognition with hidden Markov models. Trans Pattern Anal Mach Learn. 2006;33.
8. Marti UV, Bunke H. The IAM-database: an English sentence database for offline handwriting recognition. Int J Document Anal Recognit. 2002;5:39-46.

9. Zimmermann M, Bunke H. Automatic segmentation of the IAM off-line database for handwritten English text. In: 2002 International Conference on Pattern Recognition. 2002 Aug;4:35-39. IEEE.
10. Youssef A. Image downsampling and upsampling methods. National Institute of Standards and Technology; c1999.
11. Hochreiter S, Bengio Y, Frasconi P, Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies; c2001.
12. Hochreiter S. Long Short-term Memory. Neural Comput. MIT Press; c1997.
13. Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks. 2005;18(5-6):602-610.
14. Graves A, Fernández S, Gomez F, Schmidhuber J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning; 2006 Jun:369-376.
15. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J. A novel connectionist system for unconstrained handwriting recognition. IEEE Trans Pattern Anal Mach Intell. 2008;31(5):855-868.