Experimenting with Assamese Handwritten Character Recognition

Jaisal Singh[1][0000−0003−1464−9663], Srinivasan Natesan[1][0000−0001−7527−1989], Marcin Paprzycki[2][0000−0002−8069−2152], and Maria Ganzha[2,3][0000−0001−7714−4844]

[1] Indian Institute of Technology, Guwahati, Assam, India
`jaisal64@gmail.com, natesan@iitg.ac.in`
[2] Systems Research Institute Polish Academy of Sciences, Warsaw, Poland
[3] Warsaw University of Technology, Warsaw, Poland

Abstract. While Optical Character Recognition has become a popular tool in business and administration, its use for Assamese character recognition is still in early stages. Therefore, we have experimented with multiple neural network architectures, applied to this task, to establish general understanding of their performance. For the experiments, we have generated an Assamese script dataset, with over 27k images. To this dataset, LeNet5, AlexNet, ResNet50, InceptionV3, DenseNet201, and hybrid LSTM-CNN models have been applied. Next, geometric transformations that included shifting image horizontally and vertically, and rotating them, were applied, to augment the available dataset. The augmented dataset was used in experiments and accuracy of applied methods has been studied. Best accuracy, obtained on the test data, reached 96.83%.

Keywords: Character recognition, Assamese language, machine learning, Convolutional Neural Networks, data augmentation.

1 Introduction

Optical character recognition (OCR) refers to the conversion of machine-printed, or handwritten, symbols into a machine-encoded text. Lot of work has been done concerning application of OCR to Western languages. Moreover, progress has been reported for handwritten Indian scripts, like Devanagari, Gurumukhi, Bengali, Tamil, Telugu and Malayalam. However, character recognition for the Assamese language, which is spoken by an estimated 25 million people, is still at a rather preliminary stage. The aim of this contribution is to follow-up results reported in [2]. This time (1) a larger dataset was prepared (27,177 in comparison to 26,707 images), (2) this dataset was further augmented by applying shift and rotate operations, and (3) additional classes of experiments have been completed (including two additional neural network architectures), to better understand the relationship between the data and performance of various (standard) models.

In this context, content of this contribution has been organized as follows. In Section 2, we start from brief summary of pertinent state-of-the-art. Interested readers should consult [2] for more details. Section 3 describes generation of the dataset, and the preprocessing steps that were applied to it. Architectures of neural network models, used in reported experiments, and details of the experimental setup are provided in Section 4. Next, Section 5 summarizes obtained experimental results. Finally, Section 6 summarizes the main findings and outlines possible future work directions.

2 Related Work

Recognition of handwritten characters poses challenges, among others, due to inconsistent skew, alignment and individual variability. Furthermore, problems are aggravated by presence of distortions, *e.g.*, smears, (partially) faded characters, etc. These challenges manifest themselves, for instance, in digitization of Indian handwritten scripts (historical documents, in particular). Here, it is worthy noting that many Indian scripts have over 500 different symbols used in their text. This count is further enhanced by different vowel modifiers, which can be used with consonants, producing a threefold combinations of consonant-vowels (only certain vowels and consonants can be joined). Furthermore, scripts like Telugu, Gujarati, Devanagari and Bangla contain additional complex symbols. These complex symbols can also use vowel modifiers, resulting in further increase in the total count of possible symbols that can be encountered within the text that is being digitized. Somewhat easier to handle are Punjabi and Tamil scripts, due to relatively small number of possible symbols (about 70 and 150 respectively).

In this context, in [14], Chaudhuri and Pal summarize various attempts to character recognition. They pointed out that the approaches can be divided into feature-based and ones that are applied to the images directly.

Among the prominent examples of work, Sukhaswamy et.al. [3] designed a Multiple Neural Network Associative Memory (MNNAM) architecture, for recognition of printed Telugu characters. In [4], Ajitha and Rao proposed a feature (extraction) based approach for recognition of Telugu characters. In [5], authors reported 96.7% accuracy for recognizing Odia characters, using Hopfield Neural Network, with zoning. For the Tamil characters, a Convolutional Neural Network with two convolutional layers, two fully connected layers, and with ReLu activation function, was applied in [6].

In [7], Indira et al. suggested the use of neural networks for recognition and classification of typed Hindi characters. The proposed technique achieved recognition rate of 76-95% for different samples. Reported results also showed that, as the number of samples used for training increases, the recognition accuracy also improves. In [8], an approach based on Deep Convolutional Neural Network has been proposed, for offline recognition of handwritten Gurumukhi characters. Using this approach, an accuracy of 74.66% was achieved. Here, no feature extraction was used.

Only a few attempts have been made towards Assamese Character recognition. Sarma et al. [9] attempted handwritten recognition with the help of template matching, the simplest method of character recognition. Since the performance of this method is very sensitive to random noises in the images, 80% accuracy was reported. Bania et al. [10] applied the zoning concept to compute diagonal features and the Grey Level Co-occurrence Matrix, to extract texture features. They also performed word recognition using feature extraction and text segmentation. The best reported accuracy, was at 90.34%. Finally, in [2] convolution neural networks have been applied to Asamese handwritten character recognition. There, the best reported accuracy was at 94.62%. Current contribution continues research found in this publication.

3 Dataset Used in the Experiments

Assamese script, is a syllabic alphabet, used in Assam and nearby regions for many languages, such as Sanskrit, Khasi, Jaintia, etc. The current form of the script has been observed in rock inscriptions dating from the 5th century A.D. The current script bears many similarities to the Bengali script.

There are few, publicly available (online), datasets of Assamese script images. Therefore, for the work reported in [2], we have created an Assamese handwritten dataset by obtaining samples from the faculty and staff of IIT Guwahati (Dataset1), and used the Tezpur University dataset (Dataset2). Moreover, for the purpose of this work, we have adapted dataset, created by S. Mandal (Dataset3). Let us now briefly describe these datasets.

Dataset1: A group of over 200 students, faculty and staff members of IIT Guwahati provided handwritten samples, each containing the 52 Assamese characters (11 vowels and 41 consonants). This group consisted of persons representing different age, education levels, and both genders, thus leading to a broad variety of writing styles. A total of 10,994 images belonged to this dataset.

Dataset2: This dataset was created with the help of the, Tezpur University created, Online Handwritten Assamese Character dataset [11] (available from the UCI Machine Learning repository [12]]). The original dataset consists of 8235 Assamese characters, collected from 45 different writers. A combined 183 symbols, consisting of 11 vowels, 41 consonants and 131 juktakkhors (conjunct consonants) have been collected from each person. However, since juktakkhors were present only within this dataset (not in Dataset1), they have not been used in the work reported here. Moreover, since in the original dataset, characters are stored by capturing movement of the pen on a digitizing tablet, they have been turned into images, to match data collected for the Dataset1.

Dataset3: This dataset was created within the context of work of Mandal [13]. It contains both characters and word samples. For the same reasons as above, word samples have not been used in our work. This dataset is also similar to the

Dataset2, as it stores information about movements of the pen on the tablet. Here, writers consisted of senior high school students from Guwahati, and adults belonging to 18-25 age group, with a minimum bachelor qualification. Again, pen movements were turned into 13342 images of vowels and consonants.

When the three datasets were combined, they contained a total of 26,707 images of consonants and vowels. It should be noted that, overall, the combined dataset was well balanced, with each character appearing almost the same number of times. Here, let us state that, while the dataset is not openly available online, any interested party can obtain it from Srinivasan Natesan, at the email listed above (list of authors of this paper).

3.1 Data preprocessing

Next, the combined dataset was preprocessed, by applying the same steps that were reported in [2]. Specifically, the following operations have been executed (interested parties should consult [2] for more details).

- Images form Dataset1 have been *binarized* using Otsu binarization. Since images from Dataset2 and Dataset3 have been created (by repeating the recorded steps) as black-and white, they did not require binarization.
- All images have been *normalized* to minimize effects of individual handwriting styles. Specifically, they were cropped and rescaled to a standard 96×96 size image.
- Smoothing, based on 3-point moving average filter method, was applied to individual strokes for images originating from Dataset2 and Dataset3.

4 Neural network architectures

In the initial work, we have experimented with four neural network architectures: (1) LeNet5, (2) ResNet50, (3) InceptionV3, and (4) DenseNet201. Since they have been already described, and since their standard versions and implementations were used, readers are asked to consult [2] for more details. With the extended dataset that was prepared for this work, we have decided to experiment with additional neural network architectures, to build a more comprehensive image of relationship between (Assamese) handwritten image recognition and popular neural network architectures. Let us now, briefly, describe the additional architectures that have been applied.

4.1 Long Short Term Memory networks

Long Short Term Memory (LSTM) networks are a class of Recurrent Neural Networks, that can handle not only single points of input, like images, but also sequences of data, like video. A standard LSTM unit comprises of a cell, an input gate, an output gate, and a forget gate. The cell can remember values over random time intervals and the three gates regulate the flow of information in the cell. LSTM models are known to perform well with predictions based on time series data.They also have the added benefit of not facing the problem of vanishing gradient. To apply LSTM to image recognition, it has been combined with Convolutional Neural Network, to form a LSTM-CNN hybrid (following ideas described in [15]).

4.2 AlexNet

CNNs have become the preferred choice for image recognition. However, as the size of datasets began to increase, there arose a need to optimize the training to cut down the training time. Here, AlexNet, comprising of 8 layers: 5 convolutional layers, and 3 fully-connected layers, solved this problem and won the 2012 ImageNet competition. What makes AlexNet different from its predecessors is:

- Nonlinearity – *relu* was used as an activation function, instead of *tanh*, leading to reduction in the training time (by a factor of six).
- Mulitple GPUs – AlexNet trained the model simultaneously on two GPUs, thus significantly reducing train times. This is particularly important when bigger models are to be trained.
- Overlapping pooling – that resulted in reduction of the size of the network.

Finally, AlexNet uses Data augmentation and Dropout layers to reduce the problem of overfitting and to increase overall model performance.

4.3 Experimental setup

The models were implemented using Python frameworks Keras and Tensorflow. In all experiments data was split into 80% for training and 20% for testing. Each result reported here is an average of 5 runs. Unless explicitly stated, experiments have been performed for different images drawn (randomly) to the training and the testing datasets.

In all experiments, training rate 0.001 and batch size 64 were used. This point deserves an explanation. While it is a well-known fact that CNN models can go through the process of hyper-parameter tuning, and this was done to some extent in the work reported in [2], this avenue was not pursued here. The reason is that, with additional data formats (shifted and rotated) the existing solution space for model tuning becomes very large. Hence we have decided to fix the learning rate and the batch size (at reasonable values) and to observe the "general behavior" of the six models under investigation. Further hyper-parameter tuning is one of directions of possible future research. However, it may not be the most fruitful and interesting, as we argue in Section 6.

5 Experimental Results and Analysis

5.1 Complete original dataset results

The first set of experiments has been conducted by applying, above described, six neural network architectures to the "complete original dataset", consisting of

Dataset1 + Dataset2 + Dataset3. Obtained results are summarized in Table 1. In the Table we represent: testing accuracy and information after how many epochs the training process was stopped, due to the lack of further progress. Moreover, to simplify the comparison, we copy the results from [2] (Table 1), available for the four architectures that were considered there. Here, we have to recognize that some results reported in [2] were obtained for different learning rates than these reported here (0.0008 for LeNet and ResNet). Moreover, all results have been obtained for different batch sizes (128 for ResNet, and 48 for the remaining three architectures). However, as stated above, our goal was to obtain the general idea on accuracy of different architectures, and to establish the "general baseline" as to what performance can be expected from application of modern neural networks directly to the problem (without feature extraction).

Table 1. Accuracy of models applied to the complete original dataset

Model	Test Accuracy	Stop epochs	Earlier results
LSTM-CNN	74.62	100	
LeNet05	85.79	70	86.25
AlexNet	**93.04**	60	
InceptionV3	92.18	50	94.09
ResNet50	92.82	70	93.55
DenseNet201	**93.08**	50	**94.26**

A number of observations can be made. First, the performance of the LSTM-CNN hybrid is a clear outlier. This model does not work for the considered task (even after substantially longer training – 100 epochs). Similarly to the results reported in [2], LeNet5 is also not competitive (its accuracy is approximately 7% worse than that of the remaining models).

The performance of the four "competitive" models is very close to each other. The difference between AlexNet, Inception, ResNet and DenseNet is less than 1%. Since they were all trained using the same learning rate and batch size, it can be conjectured that after individual hyper-parameter tuning it is possible that their performance could be improved.

Finally, obtained results are slightly worse than those reported in [2]. The difference is also of order of 1% for all models. However, it should be kept in mind that the "old" results were obtained after some hyper-parameter tuning (and on a different, slightly smaller, dataset).

Overall, somewhat contradictory to the naive expectations, extending the size of the dataset did not result in immediate performance improvement. Therefore, we have decided to further explore how the dataset can be augmented to make the models generalize (and perform) better.

5.2 Dataset with shifted images

In the second phase of our work we have decided to augmented the existing dataset with images shifted in eight directions: Top, Bottom, Left, Right, Top Left, Top Right, Bottom Left and Bottom Right, by 2 pixels. This resulted in creation of eight new datasets, with a total of $8 \times 27,177 = 190,239$ images. These image sets have been mixed with the original ones (each direction of shift separately) and then randomly split into 80-20 datasets for training and testing. Due to the extremely poor performance for the original dataset, in this round of experiments, we have skipped the LSTM-CNN model and experimented only with the remaining five architectures. The same learning rate (0.001) and batch size (64) were used. Taking advantage of the fact that the models have been already trained on the original dataset, we used standard transfer learning to reduce the training time for the shifted images. Obtained results are summarized in Table 2. Each result is an average of 5 runs for different data splits.

Table 2. Accuracy for shifted datasets

Model	Shift Left	Shift Right	Shift Top	Shift Bottom	Top Right	Top Left	Bottom Right	Bottom Left
LeNet05	79.22	79.31	79.68	78.41	80.89	79.97	79.71	79.88
AlexNet	94.44	94.58	94.65	94.55	94.69	94.85	94.71	94.77
InceptionV3	95.53	95.45	95.52	95.57	95.75	95.61	95.65	95.57
ResNet50	**95.85**	**95.78**	**95.88**	**95.81**	**96.18**	**95.99**	**95.98**	95.91
DenseNet201	95.52	95.73	**95.88**	**95.81**	95.98	95.91	95.94	**95.95**

Comparing results presented in Table 1 and Table 2 it is easy to observe that LeNet remains the least accurate of the 5 models. In all remaining cases, augmenting dataset by shifted images results in overall performance improvement. In the Table, we have marked, in bold, the best results for each shift-direction augmentation. This brings an interesting observation. AlexNet, which was one of “overall winners”, reported in Table 1, is outperformed by ResNet and/or DenseNet.

Moreover, out of the four better performing models, for all directions of image shift, the performance difference remains of the order of 1%. Combining this with the fact that, in this series of experiments, AlexNet was outperformed by ResNet and/or DenseNet, it becomes even clearer that further hyper-parameter tuning may result in change of order of best performers. However, it can be also stated that these *four models are very much comparable in their performance.*

5.3 Dataset with rotated images

Seeing that augmentation of data with shifted images helped, we have tried use augmentation with images that are rotated by one and by two degrees in both clockwise and counterclockwise direction. This brought four new datasets with a total of $4 \times 27,177 = 108,708$ images. Rotated images have been (separately) mixed with the original dataset and randomly split into 80-20 ratio for training and testing. Again, for each dataset, 5 experiments have been run, applying transfer learning to reduce training time. Average accuracy is reported in Table 3.

Table 3. Accuracy for rotated datasets

Model	Clockwise 1 degree	Counter 1 degrees	Clockwise 2 degrees	Counter 2 degrees
LeNet05	78.44	77.78	78.47	78.64
AlexNet	94.24	94.19	94.55	94.61
InceptionV3	94.25	94.41	94.67	94.77
ResNet50	95.18	95.22	**96.01**	**96.22**
DenseNet201	**95.24**	**95.28**	95.75	95.85

Here, the general pattern of results, reported above, repeats, though with some differences. Again, LeNet is not competitive. Hence, it was not be considered in what follows. Among the remaining four models. ResNet and DenseNet are the "winners". However, the performance difference between all four models remains of order 1%. Interestingly, DenseNet outperforms ResNet for datasets with images rotated by one degree, while the reverse can be observed for images rotated by two degrees. We do not have an explanation for this behavior.

5.4 Experiments with different train and test datasets

In this set of experiments we have studied interplay between various data subsets, used for training and testing. All dataset categories, listed next, start with the dataset created by combining Dataset1 + Dataset2 + Dataset3. This *raw* dataset, consists of original images without normalisation, or smoothing (where it would have been applicable), so it includes all extra empty spaces around the symbols and rough images resulting from drawing letters on the basis of pen movements, i.e. images are exactly the way they were "produced" (before any preprocessing was applied). The *normalized* dataset consists of *raw* images that have been normalized (see, Section 3.1). The *no smoothing* dataset consists of images that have been normalized, but not smoothed. The *processed* dataset refers to the dataset that has been fully prepossessed, i.e. normalised and smoothed. The *shifted* dataset consists of all eight sets, in which images have been shifted. The *rotated* dataset includes four sets of images that have been rotated. In Table 4 presented are results obtained when treating separately *raw*, *normalized*, *no smoothing* and *processed* datasets, for the four remaining models. As previously, the same learning rate, batch size and 80-20 split were applied and each result is an average of 5 runs.

As can be seen, when raw dataset is used for both training and testing, performance of all models is lacking. Interestingly, but somewhat expectedly,

Table 4. Accuracy for basic separate datasets

Train dataset	Test dataset	AlexNet	Inception	ResNet	DenseNet
raw	raw	53.82	55.84	55.18	57.92
raw	processed	68.82	66.42	64.88	66.71
processed	raw	62.64	61.49	62.79	65.18
processed	no smoothing	94.08	93.58	93.58	94.25

when models are trained on raw dataset, but applied to processed dataset (for testing), the performance improves by 10-15%. Here the ability to "capture" features of the "more general" dataset pays dividends when the model is applied to the less general (made more uniform by preprocessing) one. This can be seen further when processed dataset is used for training and applied to the raw dataset. This reduces the performance by 2-5%, depending on the model. Finally, the last line in the Table shows that smoothing has much weaker effect on the performance than normalization. Obviously, this can be related to the fact that not all images had to be smoothed (see, Section 3.1). When processed dataset is used for training and applied to the dataset that was normalized but not smoothed, the performance is similar to the one reported in Table 1. Here, as in all cases reported thus far, the performance of the four models is very close to each other. Hence, again, it can be expected that individual hyper-parameter tuning may influence the fact that AlexNet and DenseNet slightly outperformed the remaining two models.

The final group of experiments involved processed, shifted and rotated datasets, mixed with each other in various combinations and used for training and testing (as denoted in Table 5). The remaining aspects of the experimental setup have not been changed. Obtained results are summarized in Table 5.

Table 5. Accuracy for combined datasets

Train dataset	Test dataset	AlexNet	Inception	ResNet	DenseNet
processed+shifted	processed	95.49	94.93	94.79	95.55
processed	processed+shifted	93.78	93.04	93.03	93.79
processed+rotated	processed	95.98	95.18	95.28	96.08
processed	processed+rotated	93.58	94.78	93.09	93.15
processed +shifted+rotated	processed	**96.78**	95.35	95.68	96.71
processed +shifted+rotated	processed +shifted+rotated	96.47	**95.78**	**95.87**	***96.83***

As can be expected, the best results have been obtained for the largest datasets, consisting of all individual datasets combined together and then split 80-20 for training and testing (depicted in the last line in the Table). The only exception is AlexNet, which obtained its best performance when processed + shifted + rotated data was used for training and applied to data that was processed (without the remaining two datasets included). Overall, the best performance was obtained by the DenseNet (which reached 96.83% accuracy). However, again, all results obtained by the four considered models are within 1% from each other, for each dataset that was experimented with (i.e. for each line in the Table).

6 Concluding Remarks

The aim of this work was to experimentally explore performance of neural network architectures applied directly (without feature extraction) to the recognition of Assamese characters (vowels and consonants). The main lessons learned can be summarized as follows: (1) LSTM-CNN hybrid and LeNet models are not competitive for this task; (2) without hyper-parameter tuning, using generic learning rate and batch size, the remaining four CNN's (AlexNet, Inception, DenseNet and ResNet) perform very closely to each other (for all experiments that performance difference was of order of 1%); (3) augmenting the dataset by shifted and/or rotated images (and thus increasing the size of the dataset) has definite positive effect on performance; (4) conservative baseline for accuracy of image recognition, without feature extraction, can be placed at around 95%.

Obviously, further work can be pursued including, among others, the following directions: (a) tuning hyper-parameters for each of the four best models, (b) building meta-classifiers, or (c) further augmenting the training dataset, possibly by applying GANNs to generate synthetic datasets.

However, above presented data brings about a more general reflection. When machine learning is applied to "well prepared", large datasets, performance of 95% or more has been reported. However, as can be seen in Table 4, when raw images are used in training and testing, the performance drops to about 57%. This means that, from practical point of vies, this "performance" is almost useless. In other words, if a similar method was to be applied to digitization of historical manuscripts, about 40% of characters would be misrepresented.

This also brings the question of recognition words within a text, which is an even more more complex endeavour. Specifically, this would involve pipelines, where words may or may not be split into individual characters and words/characters would have to be automatically preprocessed. Next, depending on the selected approach, an appropriate classifier would have to be trained. This can be relatively easily completed in the case of structured documents (see, for instance [1]), but is much more complex for unstructured text.

Overall, it can be stated that, while the recognition of prepossessed scripts representing symbols in majority of languages is already well understood, the main research directions concern (i) recognition of characters represented as raw images, and (ii) recognition of words originating form unstructured handwritten documents.

References

1. A.Denisiuk, M.Ganzha, M.Paprzycki, K.Wasielewska-Michniewska, *Feature Extraction for Polish Language Named Entities Recognition in Intelligent Office Assistant*, Proceedings of the 55th Hawaii International Conference on System Sciences (in press)
2. D. Mangal, M. Yadav, S. Natesan, M. Paprzycki, M. Ganzha, *Assamese Character Recognition using Convolutional Neural Networks*, Proceedings of 2nd International Conference on Artificial Intelligence: Advances and Applications, Algorithms for Intelligent Systems, 2022, https://doi.org/10.1007/978-981-16-6332-1_70 (in press)
3. M.B. Sukhaswamy, P. Seetharamulu and A.K. Pujari, *Recognition of Telugu Characters Using Neural Network*, *Int. J. of Neural Systems*, **6(3)**, 317-357, 1995.
4. P.V.S. Rao and T.M. Ajitha, *Â Telugu Script Recognition – A Feature Based Approach*, *Proc. of Third Int. Conf. on DAR*, **1**, 1995
5. O.P. Jena, S.K. Pradhan, P.K. Biswal, and A.K. Tripathy, *Odia Characters and Numerals Recognition using Hopfield Neural Network Based on Zoning Feature*, *International Journal of Recent Technology and Engineering*, Vol.8, 2019
6. A. A. Prakash and S. Preethi, *Isolated Offline Tamil Handwritten Character Recognition Using Deep Convolutional Neural Network*, International Conference on Intelligent Computing and Communication for Smart World (I2C2SW), Erode, India, pp. 278-281, 2018
7. B. Indira, M.Shalini, M. V. Ramana Muthy, M. S. Shaik, *Classification and recognition of Printed Hindi Characters Using ANN*, Int.Journal Image, Graphics & Signal Processing, 6, 15-21, 2012
8. U. Jindal, S. Gupta, V. Jain, M. Paprzycki, *Offline Handwritten Gurumukhi Character Recognition System Using Deep Learning*, Advances in Bioinformatics, Multimedia, and Electronics Circuits and Signals,vol. 1064, Springer, pp.121-133, 2020
9. P. Sarma, C. K. Chourasia and M. Barman, *Handwritten Assamese Character Recognition*, IEEE 5th International Conference for Convergence in Technology (I2CT), Bombay, India, pp. 1-6, 2019
10. R.K. Bania, R. Khan, *Handwritten Assamese Character Recognition using Texture and Diagonal Orientation features with Artificial Neural Network*, International Journal of Applied Engineering Research, 13 (10), 7797-7805, 2018
11. U. Baruah and S. M. Hazarika, *A Dataset of Online Handwritten Assamese Characters*, Journal of Information Processing Systems, 11 (3), pp.325-341, 2015; DOI: 10.3745/JIPS.02.0008
12. D. Dua and C. Graff, *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml], Irvine, CA: University of California, School of Information and Computer Science, 2019
13. S. Mandal, *Noval Approaches for Basic Unit Modeling in Online Handwritting Recognition*, PhD Thesis, Department of Electronics and Electrical Engineering, Indian Institute of Technology, Guwahati, India, 2019
14. U. Pal, B.B. Chaudhuri, *Indian script character recognition: a survey, Pattern Recognition*, vol. 37, Issue 9, 2004, pp. 1887-1899,ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2004.02.003
15. J. Zhang, Y. Li, J. Tian and T. Li, *LSTM-CNN Hybrid Model for Text Classification*, 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018, pp. 1675-1680, doi: 10.1109/IAEAC.2018.8577620