JOIV : Int. J. Inform. Visualization, 7(3) - September 2023 984-990

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Ranjana Script Handwritten Character Recognition using CNN

Jen Bati [a,*], Pankaj Raj Dawadi [a]

[a] *Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Kavre, 45210, Nepal*

Corresponding author: [*]jen.bati16@gmail.com

***Abstract*— This paper proposes a public image database for Ranjana script Handwritten Character Datasets (RHCD), publicly available for Ranjana script researchers or anyone interested in the subject. To the best of our knowledge, the Ranjana script Handwritten Character Dataset (RHCD) is the first publicly available database for Ranjana script researchers. Ranjana script descended from the Brahmi script, consists of 36 consonant letters, 16 vowel letters, and 10 numerical letters. The focus of this research is three-fold: the first is to create a new database for Ranjana script Handwritten Character Recognition; the second is to test the character recognition accuracy of the created RHCD using existing CNN algorithms like LeNET-5, AlexNET, and ZFNET algorithm; the third is to propose a model by investigating different hyper-tuning parameters to improve the recognition accuracy of the created RHCD. The research method applied in this study is dataset collection, digitization & cropping, pre-processing, dataset splitting, data augmentation, and finally, implementing the CNN model (existing and proposed). Performance evaluation is based on the test accuracy, precision, recall, and F1-score. The experiment result shows that our model ranks first, with a testing accuracy of 99.73% for 64x64 pixels resolution with precision, recall, and F1-score value 1. Creation and recognition of Ranjana script characters, vowel modifiers, and compound characters can be the next milestone to be achieved. Segmentation of words and sentences into characters and recognizing each character individually can be the next research domain.**

***Keywords*— Ranjana script; Newari script; RHCD; hyper-parameter; handwritten characters.**

Manuscript received 28 Mar. 2023; revised 10 May 2023; accepted 25 Jun. 2023. Date of publication 10 Sep. 2023.

I. INTRODUCTION

Character recognition and segmentation for Latin, Chinese, Arabic, and Devanagari scripts have been studied for many years and advanced to a certain level. However, the same has not been done in the case of the Ranjana script. Many books are written in Ranjana script on diverse topics like Science, Astrology, Medicine, History, Architecture, and Tantrism. Consider the abundant knowledge we would get if we could read the material in our preferred languages. The younger generation is not interested in studying their mother tongue, language, and script due to the lack of adaptability in the teaching methodology and resources. So, to close this gap and create more accessible platforms for the upcoming generation, digitization of this precious manuscript is in high demand.

Character Recognition tasks require a large set of reliable datasets to apply recognition algorithms and generate efficient models. In the case of the Ranjana script, there is no such character dataset for character recognition, so further research is required in this area. Therefore, this research proposes an image database for RHCD. We believe that RHCD is the first publicly accessible database for anyone working on or interested in Ranjana scripts.

A script refers to a collection of characters. Languages worldwide are written using a script, like Latin, Chinese, Arabic, Devanagari, and Ranjana script. A script is also called lipi and in the case of Newari lipi, there are 9 variations, namely Ranjana lipi, Prachalit lipi, Bhujinmol lipi, Golmol lipi, Kunmol lipi, Kwanmol lipi, Pachumol lipi, Hinmol lipi and Litumol lipi. Among them, Ranjana lipi is the most popular one. One of the several alphabets descended from the Brahmi script is the Ranjana script, also known as Kutila or Lantsa [1]. Ranjana script was developed during the 11th century and was used for writing the Newari languages. However, it is also used in Buddhist temples of Nepal, India, Tibet, China, Mongolia, and Japan. On many prayer wheels of different sizes, the well-known mantra "Om mani Padmee hum" is typically written in the Ranjana script, and in many Tibetan temples between Ladakh and Mongolia, the vibrant ceiling beams are embellished with gold-lettered mantras in the same font [2].

Like the Devanagari script, Ranjana script has 36 consonant letters, 16 vowel letters, and 10 numerals, typically

written in a left-to-right fashion. However, the Kutakshar form, which is another form of Ranjana script, is written from top to bottom. The Ranjana script is used to print Newari newspapers, wedding cards, and certificates. Even the local municipality of Kathmandu, Lalitpur, and Bhaktapur use the Ranjana script in signboards and letter pads.

Fig. 1 Ranjana script Consonants

Fig. 2 Ranjana script Vowels

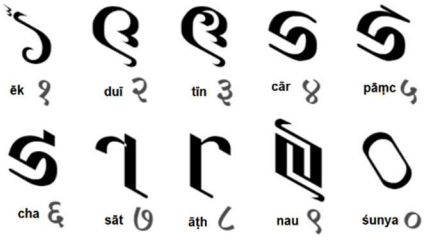

Fig. 3 Ranjana script Numerals

Many OCR technologies belonging to Latin script can provide close to 100% accuracy in recognizing text. Devanagari Optical Character Recognition is one such area that has seen many investigations. There are some publicly available benchmarks handwritten character databases available for scripts like Odia [3], Arabic [4] [5], Malayalam [6] [7], Meitei Mayek [8], Farsi [9], Telugu [10], Urdu [11], Tifinagh [12], Thai [13] and MNIST [14] datasets. However, no research has been carried out on character recognition of the Ranjana script.

Acharya et al. [15] presented DHCD, a massive handwritten Devanagari character recognition system based on deep learning. Their research shows that using dropout and increasing the dataset can prevent overfitting and increase test accuracy by up to 98.47%. Avadesh and Goyal [16] demonstrated optical character recognition for Sanskrit that can digitize dirty and neglected documents using convolutional neural networks. With an accuracy of 93.32%, they discovered that CNN is more suited for multiclass classification problems than SVMs and ANNs.

Deep convolutional neural networks were used to recognize Hindi handwritten characters [17]. When compared to LeNET-1, they discovered that LeNET-5 had an improved accuracy of 95.72%. A deep learning method for optical character recognition of handwritten Devanagari script was presented by Dessai and Patil [18]. They achieved 89.34% accuracy, and when the characters (ga), (ana), and (sa) were excluded, their accuracy went up to 91.11%.

Aneja and Aneja [19] presented a transfer learning approach to recognize Devanagari alphabets in which various pre-trained networks such as VGG 11, VGG 16, VGG 19, AlexNet, DenseNet 121, DenseNet 201, and Inception V3 are implemented. The highest accuracy of 99% is achieved using the Inception V3 model due to different regularization techniques. Ghimire et al. [20] experimented with analyzing and recognizing handwritten Nepali characters using CNN with the help of template matching technique. Their experiment concluded that the convolution neural network model has more accuracy than the Feed Forward neural network in character recognition.

Convolutional neural networks were employed by [21] to recognize handwritten characters. Utilizing 8 layers of CNN classification is carried out (4 conv, 2 pooling, 2 fully connected). 99.23% accuracy was the result. Sachdeva and Mittal [22] showed machine learning-based handwritten offline Devanagari compound character recognition. According to their research, adding more convolution layers and filters yields a testing accuracy that is 99.88% accurate.

DeepNet Devanagari, a deep learning model for recognizing ancient Devanagari characters, was introduced by Kumar et al. [23]. With an accuracy of 93.73%, the CNN algorithm was discovered to perform better than the results suggested by employing Naivs Bayes, SVM, and Decision Tree to recognize old Devanagari manuscripts. The recognition of handwritten Devanagari characters using a convolutional neural network was presented by [24]. They achieved a consonant accuracy of 96.86%, a vowel accuracy of 99%, and a numeric accuracy of 99.29%.

II. Materials and Method

A. Data collection

To build the Ranjana script handwritten characters' database, A4-sized empty data sheets were used, in which 66 pre-specified rectangle boxes were created, as shown in Figure. 4 (a). There are additional boxes for writers to note

their names on the form. Figure 4 (b) shows a sample form with 36 consonants, 16 vowels, and 10 numbers printed as a reference for the writers. The writers were instructed to write each letter in empty blocks following the reference sample form. The writers were given the forms, fountain pen, and black ink. However, some individuals used ink colors other than black too.

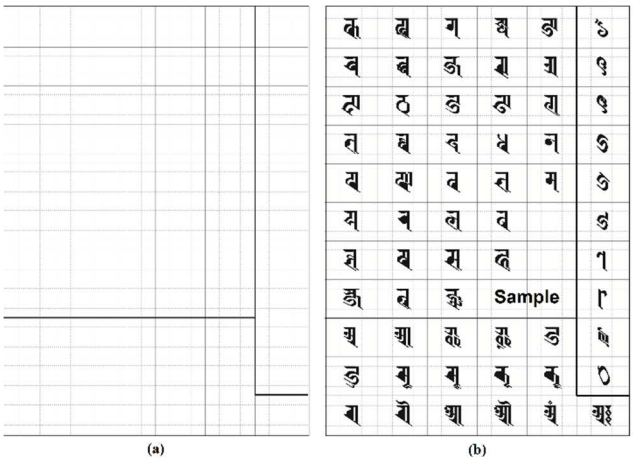

Fig. 4 A4 size blank and sample datasheet

Because of its curved nature, Ranjana script characters should be handwritten. Therefore, a special type of pen was needed, as shown in Figure 5, which was made using the following steps.

- Figure 5 a. Get an ordinary fountain pen.
- Figure 5 b. Cut the tip of the fountain pen nib with an RJ-45 clamper at a certain angle (not more than 30 degrees).
- Figure 5 c. Rub the nib on a knife sharpener stone to flatten the rough edges.

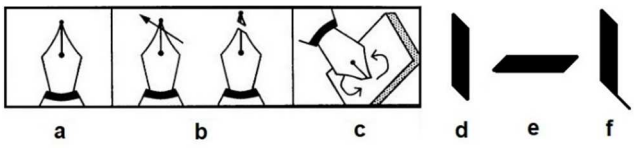

Fig. 5 Cutting Fountain Pen Nib

While writing softly on the paper, what this pen does is as follows:

- Figure 5 d. It writes with a thick lining if drawn from top to bottom.
- Figure 5 e. It writes with a thick lining if drawn from left to right.
- Figure 5 f. It writes with fine lining if drawn from left to right obliquely.

The datasheet form was distributed among the Ranjana scriptwriting community, like Nepal Lipi Guthi, Asa Safu Kuthi, and Callijatra team. The form was also distributed to the participants of a training program in Ranjana script conducted by the Rotaract Club of Baneshwor Royal. Besides these, Khwopa College of Engineering students and staff familiar with the Ranjana script also contributed to the data collection process.

Each character (36 consonants, 16 vowels, and 10 numerals) was written 1 to 2 times by each writer. Almost all the letters were written with the provided calligraphy pen and black ink, while some were written with a normal pen for different character variations. About 150 people have contributed their handwriting to the database's development.

B. Digitization of data

Submitted papers were scanned at 300 dpi resolution with a Canon MF4320-4350 scanner to digitize the collected data. The scanner was adjusted to produce PNG format directly. Before scanning, certain noisy and missing character images were manually corrected.

C. Cropping

Cropping was applied to each A4 size datasheet to crop the character blocks. The Microsoft Office Picture Manager and Adobe Photoshop CS6 were manually used to complete this operation. All similar letters were grouped in a folder, resulting in 62 folders. Each folder has 280-character images.

D. Pre-processing

All the cropped letter images were pre-processed to remove noise, resized, cropped, and edges were also detected. Pre-processing steps applied to the image are described below: -

1) Step-1 RGB to grayscale conversion: Some datasets were written in ink that was not black, such as blue and red. As it is challenging to process color images, the first step is to convert the image to grayscale. Python OpenCV's cv2.cvtColor() techniques were used for this. A sample of a Grayscale image is shown in Figure 6. For the RGB image f(x,y), the following equation gives the corresponding grayscale image.

$$g(x,y) = 0.2989 * R\big(f(x,y)\big) + 0.5870 * G\big(f(x,y)\big) + 0.1140 * B\big(f(x,y)\big) \quad (1)$$

were, R(f(x,y)) is red component of the RGB image f(x,y), and so on.

Fig. 6 Grayscale image

2) Step-2 Grayscale to binary conversion: Turning a grayscale image into a black-and-white image is called image binarization (i.e., 0 and 255 pixels, respectively). The grayscale image is then transformed into a binary image using the binary thresholding Python OpenCV procedure, where Otsu thresholding is used as an additional flag. If the image intensity is less than a threshold value T, it is made black (0) or white (255). A sample of a binary image is shown in Figure 7.

$$P(x) = \begin{cases} 0, x < T \\ 1, x \geq T \end{cases} \quad (2)$$

where P(x) is the newly updated pixel value, and x is the original pixel value.

Fig. 7 Binary image

3) Noise removal: Pre-processing is mostly used to get rid of image noise in all its forms and increase character image correctness. Filtering aids in the removal of any extraneous pixels. To eliminate background noise, Python OpenCV Morphological procedures were utilized. In Figure 8, a sample of the filtered image is displayed.

Fig. 8 Filtered image

4) Universe of Discourse: It is essentially the smallest rectangle surrounding the character and is the smallest space in which the character is present. The excess undesirable pixels around the image are eliminated. As illustrated in Figure 9, the borders in each image were located using the Python OpenCV findContours() approach.

Fig. 9 Universe of Discourse

5) Normalization: Normalization reduces a variable-sized input image to a fixed size that can be processed. For our work, we kept the original image pattern while scaling the input images to 64 x 64 pixels (width: 64, height: 64). A sample of a normalized image is shown in Figure 10.

Fig. 10 Normalized 64x64 pixel

6) Inversion: We used white pixels as the foreground and black pixels as the background for character identification. Therefore, it is necessary to invert the images. We used Inverse-Binary Thresholding, which converts the black and white pixels into white, respectively. Figure 11 displays an image that has been reversed.

$$i(x, y) = 1 - f(x, y) \quad (3)$$

Fig. 11 Inverted image

E. Training/Validation/Testing set split

We used a split ratio of 60:20:20 to randomly divide the dataset into the three subsets known as the Training, Validation, and Test set. To train the model, a training set is utilized. A validation set is used to validate the model performance during training. The test set is used to test the model after completing the training. The training folder contains 60% (10,416-character images, i.e., 168-character images per class), the validation folder contains 20% (3,472-character images, i.e., 56-character images per class), and the testing folder also contains 20% (3,472-character images, i.e., 56-character images per class) of the total character images.

F. Data Augmentation

Techniques like data augmentation can be applied to get over the issue of a smaller number of datasets and attain high accuracy[23]. Data augmentation is required because the dataset used in this study is in few quantities. Each character image is augmented 10-fold using Keras ImageDataGenerator subjected to a random amount of rotation_range = +-20°, zoom_range = 0.20, and shear_range = 0.50. The database can be downloaded at: https://kaggle.com/datasets/a1a3bb5d8fc063cbed7fba4d5662df60b63c501be5766d30ef1dc7a441ef8fdb.

G. Proposed CNN Architecture

The proposed architecture's structure is [CONV-CONV-POOL - CONV-CONV-POOL - CONV-CONV-POOL - FC-FC-FC-FC]. The architecture detail is given in Table 1 with trainable and non-trainable parameters. The proposed model is used for 64x64 pixels input image. Hyper-parameters tuning using Keras tuning was done in order to obtain the best hyper-parameters for the proposed model.

TABLE I
PROPOSED ARCHITECTURE

Input image 64x64 pixels	
Layers	**Param**
Conv2D 16 3x3	160
Conv2D-32-5x5	12832
BN, Maxpool (dp 0.45)	128
Conv2D-64-3x3	18496
Conv2D-64-5x5	102464
BN, Maxpool (dp 0.40)	256
Conv2D-128-5x5	204928
Conv2D-64-3x3	73792
BN, Maxpool (dp 0.45)	256
Fc-512	2097664
BN (dp 0.50)	2048
Fc-512	262656
BN (dp 0.45)	2048
Fc-64	32832
BN (dp 0.35)	256
Fc-62	4030
Total params: 2,814,846	
Trainable params: 2,812,350	
Non-trainable params: 2,496	

The relu activation function is found to be the best activation function by tweaking the activation function. By tuning the optimizers, adam optimizer is obtained as the best optimizer. By tuning the learning rate, 0.000775 learning rate is obtained as the best learning rate. By tuning the batch sizes, the highest score for the batch size obtained is 32.

III. Results and Discussion

A laptop with an Intel Core i7-10750H 2.60 GHz (CPU), 16GB of RAM, and an NVDIA GeForce GTX 1650 Ti VGA card was used to conduct our research. We also used Kaggle Notebook for a session with a maximum of 73.1 GB disk, 13 GB of RAM, and 15.9 GB of GPU memory. Tensorflow and Keras were used with Python 3.9.12 and Windows 11 to build the system.

The collected Ranjana script Handwritten Character Datasets (RHCD) contain a wide range of variations in the characters, including thickness, color, size, and style of the characters, this variation will facilitate in increasing the character recognition accuracy of Ranjana script in natural images. However, extreme variations or non-standard character designs can pose challenges and reduce recognition accuracy. Therefore, to optimize character recognition, we have recommended following the sample datasheet shown in Figure 4 (b), containing standard fonts, consistent letter sizes, and legible styles to maintain data consistency during the dataset collection process.

We chose three well-known CNN architectures — LeNET-5 [25], AlexNET [26], and ZFNET [27] — as well as an extra proposed design to demonstrate the value of our proposed dataset for research in RHCD. We have looked at several hyper-tuning parameters when putting the suggested architecture into practice to get the best performance outcomes in terms of test accuracy, precision, recall, and f1-score. The accuracy rate comparison was completed using current CNN architectures, and a suggested CNN design (see Fig. 12).

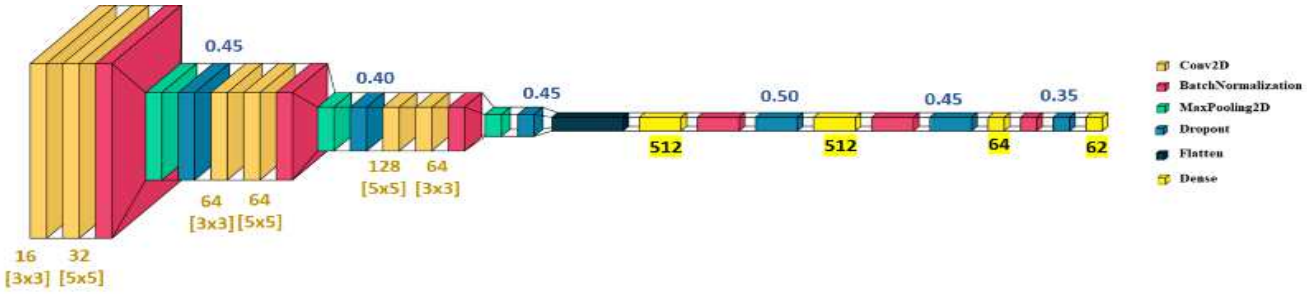

Fig. 12 Proposed CNN architecture

We have avoided using high-performance CNN such as VGGNET [28], GoogLeNet [29], and Microsoft Resnet [30] because they require huge computation power, which is not available in our ordinary computers and laptops. For input image size 64x64 pixels resolution, an accuracy of 88.06% is achieved with LeNET5 architecture at 56 epochs. The experiment was then performed with AlexNET architecture, and an improvement in accuracy up to 98.90% was observed, with the number of epochs being 53 epochs. The experiment was then performed with ZFNET architecture, and 98.35% accuracy was observed, with the number of epochs being 8 epochs. The experiment was then carried out using the proposed architecture, which had the maximum accuracy rate of 99.73%, with precision 1, recall 1, and F1-score 1 which is obtained at 53 epochs.

TABLE II
Accuracy Result of Various Architecture

Input image 64x64 pixels					
CNN	**Test Accuracy**	**Precision**	**Recall**	**F1-score**	**Epoch**
LeNET - 5	88.06%	0.97%	0.86%	0.86%	56
AlexNET	98.90%	0.98%	0.98%	0.98%	53
ZFNET	98.35%	0.97%	0.97%	0.97%	8
Proposed model	99.73%	1	1	1	53

Figure 13 shows Training/Validation accuracy curve for 64x64 pixels resolution for the proposed model at different epochs. The proposed model shows low bias and low variance, resulting in the best fit curve.

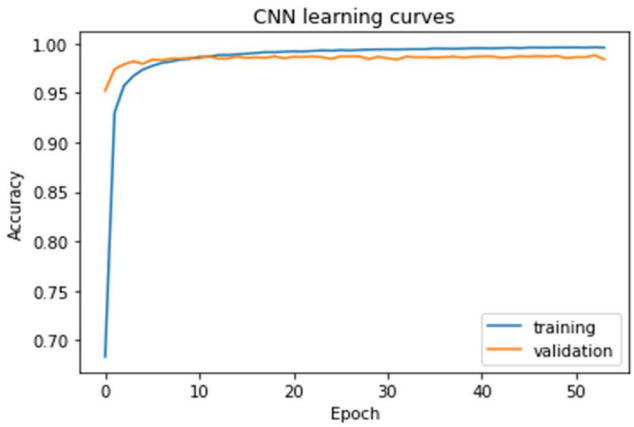

Fig. 13 Training/Validation accuracy curve for the proposed model

Figure 14 shows the Training/Validation loss curve for 64x64 pixels resolution for the proposed model at different epochs.

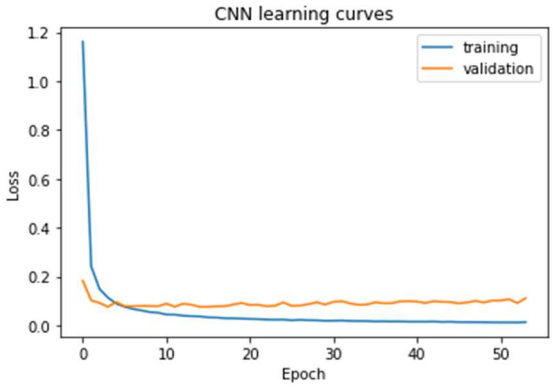

Fig. 14 Training/Validation loss curve for the proposed model

Figure 15 represents the confusion matrix generated by the proposed model for 64x64 pixels input image size with a maximum test accuracy of 99.73%. The diagonal line represents the number of characters correctly classified into their respective classes. Moreover, the values other than the diagonal line are those misclassified values. The misclassification is due to the similarity between some characters like (ba and va), (4 and 5), (2 and 3), (lirr and lir) (chha and tha).

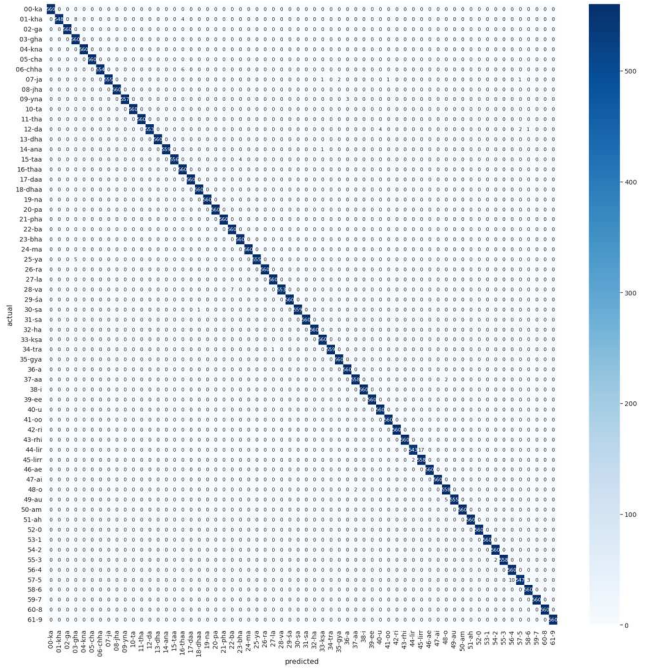

Fig. 15 Confusion matrix generated by the proposed model

IV. Conclusion

Any recognition system must start with a standard image database. In this research, we propose a new database for the Ranjana script Handwritten Character Dataset, which will be open to all researchers. To the best of our knowledge, this Ranjana script Handwritten Character Dataset (RHCD) is the first database of its kind that is freely accessible to researchers. There are 173,600 character images in the database, divided into training, validation, and testing folders, which are again subdivided into 62 characters.

We evaluated the created Ranjana script Handwritten Character Dataset (RHCD) using Le-NET-5, AlexNET, ZFNET, and a proposed CNN model architecture. The evaluation was performed for 64x64 image pixels resolution. Hyper-parameter tuning using the Keras tuner was done for finding the hyper-parameter values. Hyper-parameter tuning was done to find the number of filters, filter size, dense layer unit, dropout values, activation function, optimizers, learning rate, and batch sizes. The proposed architecture's best accuracy rate was a test accuracy of 99.73% for 64x64 pixel resolution.

Hyperparameter tuning may result in identifying optimal hyperparameters by finding the right balance between the number of layers, filter sizes, pooling sizes, and batch sizes, the model can achieve a good accuracy rate even with a less complex model like ours while being computationally efficient.

In the future, we would like to increase the size of the dataset and apply various other models to achieve a higher accuracy rate. Creating and recognizing Ranjana script characters with vowel modifiers can be the next milestone. Again, the creation and recognition of Ranjana script compound characters can be the next future research work. Segmentation of words and sentences into characters and recognizing each character individually can be the next research domain.

Acknowledgment

We thank Nepal Lipi Guthi, Asa Safu Kuthi, the Callijatra team, the Rotaract Club of Baneshwor Royal, Khwopa College of Engineering, as well as every single person who volunteered readily in contributing handwriting samples for the preparation of the database.

References

[1] E. R. Acharya, "Ranjana Numeral System: A Brief Information," *Journal of the Institute of Engineering*, vol. 13, no. 1, pp. 221–224, Jun. 2017, doi: 10.3126/JIE.V13I1.20370.

[2] Jens-Uwe and Hartmann, "The Ranjana Script," 1998, Accessed: May 11, 2023. [Online]. Available: https://epub.ub.uni-muenchen.de/25506/1/Hartmann_Ranjana_Script.pdf.

[3] R. K. Mohapatra, T. K. Mishra, S. Panda, and B. Majhi, "OHCS: A database for handwritten atomic Odia Character Recognition," 2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), Dec. 2015, doi: 10.1109/ncvpripg.2015.7490020.

[4] M. N. AlJarrah, M. M. Zyout, and R. Duwairi, "Arabic Handwritten Characters Recognition Using Convolutional Neural Network," 2021 12th International Conference on Information and Communication Systems (ICICS), May 2021, doi: 10.1109/icics52457.2021.9464596.

[5] A. A. A. Ali and S. Mallaiah, "Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 6, pp. 3294–3300, Jun. 2022, doi: 10.1016/j.jksuci.2021.01.012.

[6] Jabir Ali V, Joseph JT. A convolutional neural network based approach for recognizing malayalam handwritten characters. Int J Sci Eng Res. 2018.

[7] K. Manjusha, M. A. Kumar, and K. P. Soman, "On developing handwritten character image database for Malayalam language script," Engineering Science and Technology, an International Journal, vol. 22, no. 2, pp. 637–645, Apr. 2019, doi: 10.1016/j.jestch.2018.10.011.

[8] D. Hijam and S. Saharia, "On developing complete character set Meitei Mayek handwritten character database," *Visual Computer*, vol. 38, no. 2, pp. 525–539, Feb. 2022, doi: 10.1007/S00371-020-02032-Y.

[9] Y. A. Nanehkaran, J. Chen, S. Salimi, and D. Zhang, "A pragmatic convolutional bagging ensemble learning for recognition of Farsi handwritten digits," *Journal of Supercomputing*, vol. 77, no. 11, pp. 13474–13493, Nov. 2021, doi: 10.1007/S11227-021-03822-4.

[10] R. Kummari and C. Bhagvati, "UHTelHwCC: A Dataset for Telugu Offline Handwritten Character Recognition," *Communications in Computer and Information Science*, vol. 1576 CCIS, pp. 249–262, 2022, doi: 10.1007/978-3-031-07005-1_22.

[11] K. O. Mohammed Aarif and P. Sivakumar, "Multi-Domain Deep Convolutional Neural Network for Ancient Urdu Text Recognition System," Intelligent Automation & Soft Computing, vol. 33, no. 1, pp. 275–289, 2022, doi: 10.32604/iasc.2022.022805.

[12] L. Niharmine, B. Outtaj, and A. Azouaoui, "Tifinagh handwritten character recognition using optimized convolutional neural network," International Journal of Electrical and Computer Engineering (IJECE), vol. 12, no. 4, p. 4164, Aug. 2022, doi: 10.11591/ijece.v12i4.pp4164-4171.

[13] A. Onuean, U. Buatoom, T. Charoenporn, T. Kim, and H. Jung, "Burapha-TH: A Multi-Purpose Character, Digit, and Syllable Handwriting Dataset," Applied Sciences, vol. 12, no. 8, p. 4083, Apr. 2022, doi: 10.3390/app12084083.

[14] A. Sharma, H. Bhardwaj, A. Bhardwaj, A. Sakalle, D. Acharya, and W. Ibrahim, “A Machine Learning and Deep Learning Approach for Recognizing Handwritten Digits,” Computational Intelligence and Neuroscience, vol. 2022, pp. 1–7, Jul. 2022, doi: 10.1155/2022/9869948.

[15] S. Acharya, A. K. Pant, and P. K. Gyawali, “Deep learning based large scale handwritten Devanagari character recognition,” 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Dec. 2015, doi: 10.1109/skima.2015.7400041.

[16] M. Avadesh and N. Goyal, “Optical Character Recognition for Sanskrit Using Convolution Neural Networks,” 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), Apr. 2018, doi: 10.1109/das.2018.50.

[17] D. Chaudhary and K. Sharma, "Hindi Handwritten Character Recognition using Deep Convolution Neural Network," *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2019, pp. 961-965.

[18] B. Dessai and A. Patil, “A Deep Learning Approach for Optical Character Recognition of Handwritten Devanagari Script,” 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Jul. 2019, doi: 10.1109/icicict46008.2019.8993342.

[19] N. Aneja and S. Aneja, “Transfer Learning using CNN for Handwritten Devanagari Character Recognition,” 2019 1st International Conference on Advances in Information Technology (ICAIT), Jul. 2019, doi: 10.1109/icait47043.2019.8987286.

[20] A. Ghimire, A. Chapagain, U. Bhattarai, and A. Jaiswal, “Nepali Handwriting Recognition using Convolution Neural Network,” International Research Journal of Innovations in Engineering and Technology, vol. 04, no. 05, pp. 05–09, 2020, doi: 10.47001/irjiet/2020.405002.

[21] Y. Gurav, P. Bhagat, R. Jadhav, and S. Sinha, “Devanagari Handwritten Character Recognition using Convolutional Neural Networks,” 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Jun. 2020, doi: 10.1109/icecce49384.2020.9179193.

[22] J. Sachdeva and S. Mittal, “Handwritten Offline Devanagari Compound Character Recognition Using CNN,” Lecture Notes on Data Engineering and Communications Technologies, pp. 211–220, 2022, doi: 10.1007/978-981-16-6289-8_18.

[23] S. R. Narang, M. Kumar, and M. K. Jindal, “DeepNetDevanagari: a deep learning model for Devanagari ancient character recognition,” Multimedia Tools and Applications, vol. 80, no. 13, pp. 20671–20686, Mar. 2021, doi: 10.1007/s11042-021-10775-6.

[24] I. Dokare, S. Gadge, K. Kharde, S. Bhere, and R. Jadhav, “Recognition of Handwritten Devanagari Character using Convolutional Neural Network,” 2021 3rd International Conference on Signal Processing and Communication (ICPSC), May 2021, doi: 10.1109/icspc51351.2021.9451716.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[27] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” Lecture Notes in Computer Science, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[28] K. Simonyan, A. Z. preprint arXiv:1409.1556, and undefined 2014, "Very deep convolutional networks for large-scale image recognition," *arxiv.org*, 2015, Accessed: May 11, 2023. [Online]. Available: https://arxiv.org/abs/1409.1556.

[29] C. Szegedy *et al.*, "Going deeper with convolutions," *cv-foundation.org*, Accessed: May 11, 2023. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." pp. 770–778, 2016. Accessed: May 11, 2023. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/