2024 2nd International Conference on Advancement in Computation & Computer Technologies (InCACCT) | 979-8-3503-7131-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/InCACCT61598.2024.10551027

Optical Character Recognition (OCR) in Handwritten Characters Using Convolutional Neural Networks to Assist in Exam Reader System

Lekshmy P L
Department of Computer Science and Engineering, LBS Institute of Technology for Women,
Kerala, India
lekshmyvinod@gmail.com

S. Velmurugan
Department of Electronics and Communication Engineering,T.J.S. Engineering College,
Chennai, Tamil Nadu, India
veluvsi@gmail.com

Indra Kumari
Department of Machine Learning Data Research Applied AI, Korea National University of Science and Technology , (UST)
Daejeon, South Korea
kumariindra7@gmail.com

S. Kayalvili
Department of Artificial Intelligence, Kongu Engineering College (Autonomous),
Erode, Tamilnadu, India
kayalvili.ai@kongu.edu

B. Teja sree
Department of Information Technology,S.R.K.R. Engineering College,
Chinaamiram, Bhimavaram, Andhra Pradesh, India
tejasree9479@gmail.com

P. Karthik Kumar
Department of Computer Science and Engineering (Cyber Security), Karpagam College of Engineering,
Coimbatore, India
karthikkumarphd@gmail.com

***Abstract*- This work aimed to develop a character recognition method to facilitate the correction of answer cards in the Multiprova software through the development of a response card analysis flow that would culminate in the recognition of letters written by students and automatic correction of the tests. To isolate and identify the answers written on the answer cards, image segmentation techniques were used based on fixed marks printed on the cards. To recognize the letters and numbers written on the cards, trained three convolutional neural networks (for digits, letters and true or false). The results achieved (98.84% accuracy for digit CNN, 98.38% accuracy for letter CNN and 99.89% accuracy for true or false CNN) point out a great average success rate.**

Keywords- Character recognition, Convolutional Neural Networks (CNNs), Exam reader systems, Image preprocessing, Error detection, Correction assistance

I. INTRODUCTION

Due to the digitization of educational materials, the way exams are administered and scored has changed dramatically [1]. However, even with the advantages of digital systems, character recognition errors can still occur - especially with handwritten responses [2]. Due to these errors, test reading systems may become less accurate and efficient, requiring human intervention to correct them [3-4].

To aid the correction process, this study proposes the integration of convolutional neural networks (CNN) into exam reading systems [5]. CNNs are a good choice for handwritten receipt character recognition as they have achieved impressive success in image recognition tasks [6]. The proposed solution uses CNN to correct misrecognized characters and perform debugging automatically [7]. A feedback loop of data collection, preprocessing, model training, character recognition, error detection, proofreading assistance, manual review and continuous improvement are the steps in the multi-step process used by the system [8]. Significantly simplifying the debugging process of this integration saves time and provides a better user experience [9]. Ultimately, by harnessing the power of CNNs, exam reader systems can achieve higher levels of efficiency and accuracy in character recognition and correction [10].

The paper is organized as review part, methodology, results and followed by conclusion.

II. LITERATURE REVIEW

The paper proposes a method to convert handwritten Bengali characters to digital format [11]. This method could pave the way for future research and practical applications. The experimental data is sourced from the isolated BanglaLekha dataset [1]. By utilizing a convolutional neural network, the model attained an accuracy of 91.81% for the 50 character classes in the base dataset. Through data augmentation with 200,000 additional images, the test set accuracy improved to 95.25%.

Another paper [12] introduces a memory-based associative model, the Hopfield network, as a fully connected layer for storing classification models in CNN architectures like LeNet-5. To evaluate the performance of this new architecture, the Odia character dataset is compared with other classification models.

A study [13] investigates the effectiveness of two different models, the sequence annotation model and the Seq2Seq NN model based on recurrent neural networks, in detecting English grammatical errors (EGE). It proposes an EGE DAC method based on sequence annotation using the new model. Additionally, it suggests a method combining the sequence annotation model and the Seq2Seq NN model, capable of handling various types of EGE. This approach considers common confusions in grammatical errors, enhancing its accuracy.

In another paper [14], the authors propose a state-of-the-art machine learning algorithm for selecting appropriate words, crucial for lexical substitutions (LS) and grammatical error correction (GEC). They demonstrate the benefits of using bidirectional long-term memory (LSTM) markers. Unlike traditional approaches, this unsupervised method relies solely on a high-quality text corpus, demonstrating

979-8-3503-7131-4/24/$31.00 ©2024 IEEE

superior performance in both domain-specific and general writing tasks.

Lastly, a study [15] offers convincing evidence for the validity of the Questionnaire of English Writing Self-Efficacy (QEWSE). Through comprehensive validation from multiple sources, including confirmatory factor analysis, it demonstrates strong content support and internal structure validity. Significant correlations between self-efficacy scores, writing ability, and self-regulated learning strategies further validate the QEWSE.

III. METHODOLOGY

A. Programming languages

In this work, two programming languages were used. Python used to process the response card images and create and train the neural network. The second, the Javascript language, in carrying out the porting of the trained network configurations to the mobile environment in order to predict the students' responses on the cards.

B. Exam Reader System

The Exam Reader software comprises two main components: the client side (front-end) and the server side. The client-side consists of two interfaces: one for browsers (desktop and mobile) and a native mobile application. The browser application, developed using React, serves as the primary platform for students and teachers to interact. It facilitates the entire process of virtual tests, including test creation by teachers and execution by students.

C. Datasets

For both cases (numbers and letters) some students from the School of Science and Technology, were asked to complete a card with examples of the required characters. An example of this completed card can be seen below in Figure 1.

Fig. 1. Card with samples made by students.

Even though each student made an equal number of characters for each class, the number of images per class did not remain the same. This happened due to filtering done on the character samples. If the figure was not of good quality, it was discarded, which is why some classes have more images than others. For all datasets (letters, digits and V or F) the number of samples from the classes were unbalanced and for better training of neural networks the number of samples per class must be balanced. It is possible to observe the imbalance of the classes in Figures 2, and 3.

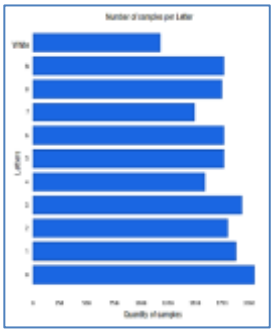


Fig. 2. Number of samples per class (digits).

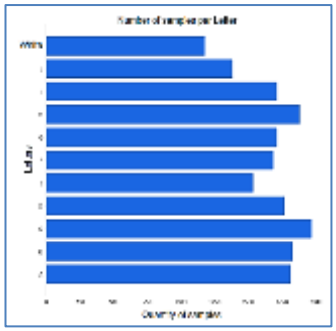


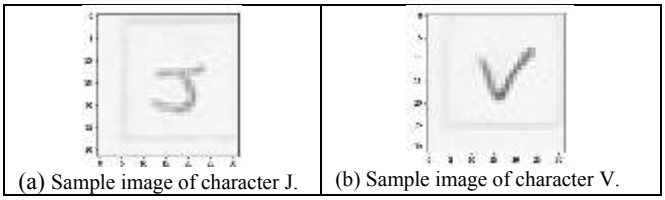
Fig. 3. Number of samples per class (letters).

This work employed data balancing techniques to ensure a balanced distribution of samples per class, utilizing two main techniques: oversampling and undersampling. Oversampling addressed situations where certain classes had significantly fewer samples compared to others by generating new samples for the underrepresented class through modifications to existing images. By applying oversampling techniques, the dataset was effectively balanced, ensuring each class had a comparable number of samples for training the models.

IV. RESULTS AND DISCUSSION

A. Image pre-processing

Pre-processing is necessary due to the need to reduce the discrepancy in the number of samples. Generating new images using mathematical transformations based on the base samples is an effective process for increasing the diversity of data present in the dataset. As seen before, Figure 3, it is possible to observe the large difference in the number of samples per class in the letters dataset. The images have dimensions 32 x 32 x 3 (height x width x number of color channels). Some examples of image samples from classes J and V can be seen in Figure 4.



(a) Sample image of character J. (b) Sample image of character V.

Fig. 4. Base letters used in data augmentation.

After the data augmentation process, the new samples can be seen in Figures 5 and 6.

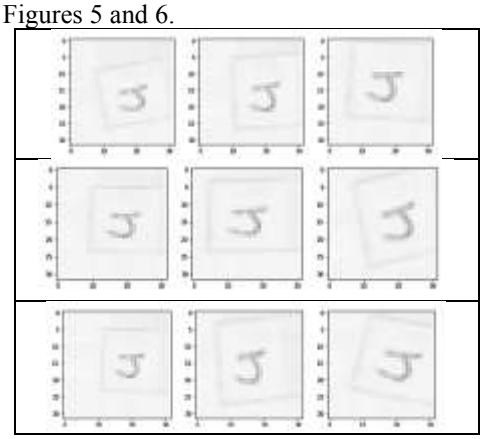


Fig. 5. Sample images of character J after data augmentation.

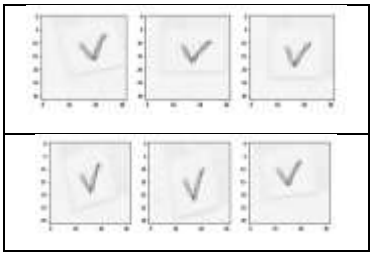


Fig. 6. Sample images of the character V after data augmentation.

Figure 5 provides an example of the base images utilized for generating new samples. In Figures 5 and 6, the outcome of each class in the data augmentation process is illustrated. Notable effects such as zooming in, zooming out, rotation, and displacement are observable compared to the original samples.

Following the balancing of the number of samples, all images per class were amalgamated into a single dataset. Initially, for the letters dataset, as depicted in Figure 3 in the previous section, the number of samples per class exhibited disparity, ranging from approximately 1100 (in the White class) to 2000 samples (in class C). This variance reflects an approximate 81% disparity in the number of images between class C and class White.

In the numbers dataset, the maximum difference in sample counts per class was approximately 90%, observed between the White class with 1100 samples and class 0 with 2100 samples. This discrepancy decreased to 20% in class 7, which had 5800 samples compared to other classes (0, 1, 2, 3, 5, 6, 8, and 9 with 7000 samples).

Such disparities in dataset image counts can lead to training network problems and potential classification errors. While the CNN may adeptly identify classes with a high number of samples, it may falter in recognizing classes with fewer samples.

Following the complete class balancing process (as depicted in Figure 7), the percentage difference in sample counts (now reduced to a maximum of 20%) serves to rectify discrepancies in sample quantities and thereby mitigate potential errors.

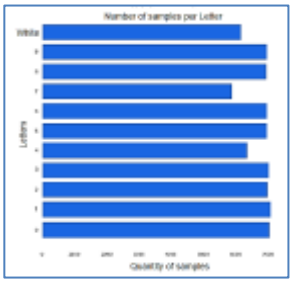


Fig. 7. Number of samples per class after increasing data (digits).

B. CNN Training

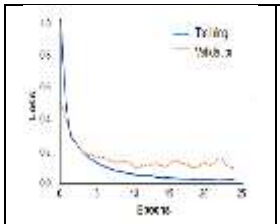
In general, CNN training occurred properly, allowing results with accuracy above 98%, and loss below 6% in its best configuration. For recognizing letters and digits, structures with 4 *convolutional* layers, followed by max pooling, proved to be more efficient. In the case of v or f character recognition, structures with 3 convolutional layers, followed by max pooling, presented the best results.

C. Digits

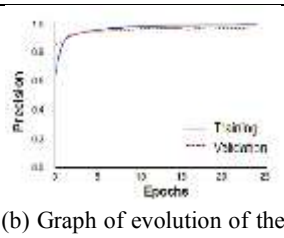
With the digits, classes with structural differences in their writing are presented, for example, 1 can be written with just one vertical line, while 8 is written with two stacked circles. As each class is considerably different from each other, it is expected that the CNN will not present difficulties in adapting to the classes during training. In Figure 7, you can see the number of samples per class, used in training the digit-specialist CNN. The number of images per class remained between 5700 and 7000, with an average of approximately 6700 samples per class. Next, the structures used in training the Digit CNN will be presented, as well as their configurations and generated results.

• Structure 1

Structure 1 also shows improvement, though not as significant as the transition to structure 2.



(a) Loss factor evolution graph during training.



(b) Graph of evolution of the accuracy factor during training.

Fig. 8. Loss and precision graphs (digits).

With an overall average accuracy of 97.8% and the worst class achieving 95% accuracy, the third structure refined results using additional convolutional and max pooling layers. Figure 8 graphs display narrowed differences between training and validation sets, with improved convergence towards appropriate values (approximately 100% for precision and 0% for loss). This enhancement is evident in the confusion matrix (Figure 9), with a maximum error rate of 3.1% and few errors surpassing 1%. Notably, classes 0 and 4 achieved 99% accuracy, underlining the model's effectiveness.

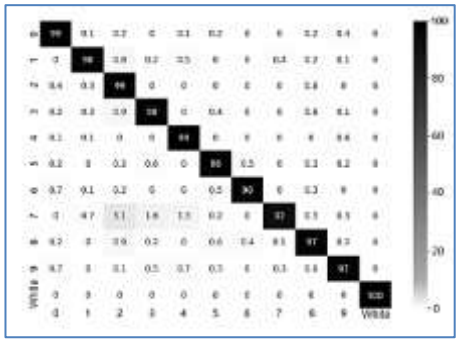
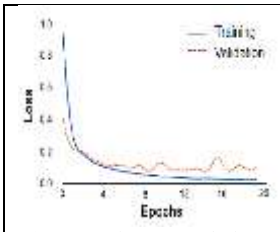


Fig. 9. Confusion matrix (digits).

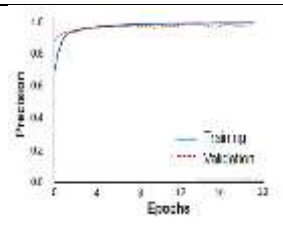
• Structure 2

Finally, Structure 2 indicates yet another small improvement in relation to Structure 1. With an overall average accuracy of 98.84% and the worst class with 98% accuracy, the fourth structure presented the best result among the 4 configurations analyzed.

The loss and precision graphs (Figure 10) do not show major differences. The oscillations in the validation curve of the loss graph (Figure 10a) are still present. Comparing the graphics with those of the previous structure, the increase in quality is not so noticeable.



(a) Loss factor evolution graph during training.



(b) Graph of evolution of the accuracy factor during training.

Fig. 10. Loss and precision graphs (digits).

In the confusion matrix - Figure 11 - it is possible to verify the improvement in the quality of the CNN. With a maximum error of 0.9%, the matrix presents few errors above 0.5% and reinforces the excellent success rate of the classes on the main diagonal (which was not less than 98%), with emphasis on class 6, which achieved 100% accuracy. For the CNN that classifies the digits, 4 CNN structures were analyzed, where the one that proved to be most efficient was the fourth (with more convolutional and maximum pooling layers). The number of epochs necessary to enable early stopping and stop training earlier remained at 24 for structures 1, 2 and 4 while Structure 1 converged faster with only 19 epochs.

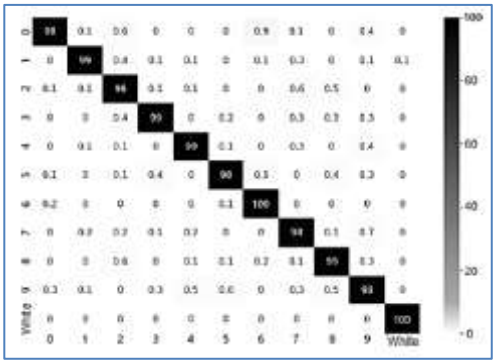


Fig. 11. Confusion matrix (digits).

Table 1 summarizes the evolution process of precision and loss of the digit CNN.

TABLE I. EVOLUTION OF THE DIGIT CNN THROUGH STRUCTURES

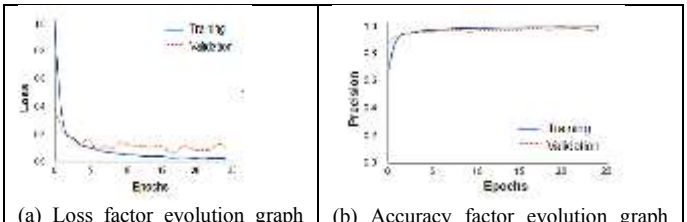
Structure	Precision	Loss
Structure 1	97.80%	0.098
Structure 2	98.84%	0.064

D. Letters

With letters, structural differences are not as discrepant as with digits. Some classes share similarities in writing, for example, classes E and F differ only by a straight horizontal line at the bottom and classes I and J have the similarity of a vertical straight line and a horizontal straight line at the bottom, even though the Class I may have another horizontal line at the top, much of the character structure is the same. As a result, a worsening in the final hit rate is expected in comparison to the digit CNN. In Figure 7 the number of samples per class used in training the letter CNN can be seen. The number of images per class remained consistently at 5000, totaling approximately 55000 figures.

• Structure 1

With 98.1% accuracy - an increase of 0.58% compared to structure 2 (a considerably smaller improvement than that achieved between structures 1 and 2 of 4.16%) - Structure 1 made good use of convolutional layers and extra maximum pooling, although the increase in precision was not large. The graph in Figure 12a shows an improvement in the convergence of the validation curve.



(a) Loss factor evolution graph during training.

(b) Accuracy factor evolution graph during training.

Fig. 12. Loss and precision graphs (letters).

Even though the oscillations during training remain present, the validation curve came closer to the ideal value and the training curve, which indicates a better adaptation of the CNN to data that it was not presented with. The graph in Figure 12b maintains the evolution perceived between structures 1 and 2, the curves converging to values closer to 1 and the reduction in the difference between the training and validation curves.

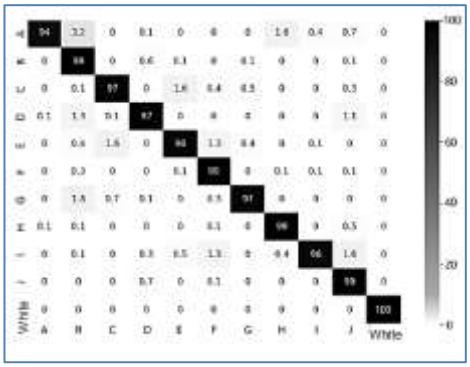
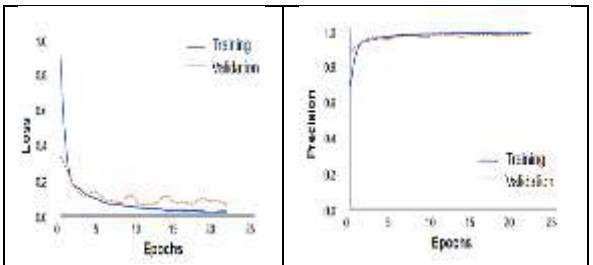


Fig. 13. Confusion matrix (letters).

In the confusion matrix (Figure 13), the general improvement is noticeable even though the worst point of confusion - in classes A and B - is worse than in structure 2 - increased from 2.7% to 3.2%. The accuracy of the worst class remained at 94% (in class A) and the best accuracy rates (with 99% accuracy) were classes B, F, H and J.

• Structure 2

Structure 2 (Figure 14) demonstrates a slight improvement over Structure 1, with an overall average accuracy of 98.38%, a mere 0.28% increase. Despite this marginal improvement, it represents the best-performing configuration among the four analyzed. Notably, the worst-performing class achieved a 96% accuracy rate. However, due to the small increase in precision, which is only 0.28%, the quality enhancement is not readily discernible. The validation curve of the loss graph exhibits fewer oscillations, and the convergence value of the training and validation curves has become closer, indicating improved stability during training. Nonetheless, when compared to Structure 1, the graphical differences are not prominently evident.



(a) Loss factor evolution graph during training

(b) Precision factor evolution graph during training

Fig. 14. Loss and precision graphs (letters).

In the confusion matrix - Figure 15 - the improvement in quality of the CNN, compared to Structure 1, is not as visible. With a maximum error of 2.3%, the matrix presents few errors above 1% and reinforces the good success rate of the classes on the main diagonal (which was no less than 96% in class D).

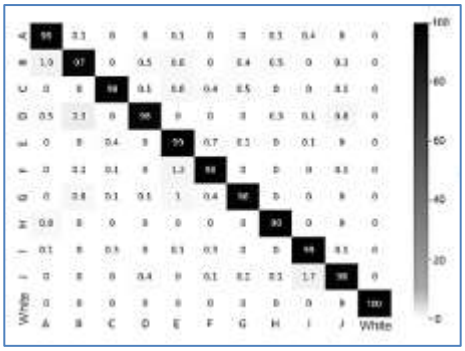


Fig. 15. Confusion matrix (letters).

Just like the neural network specialized in digits, the CNN that classifies letters was analyzed with 4 different CNN structures, where the one that proved to be most efficient was the fourth. Even though the results were good, they were not as good as CNN's single-digit results. The number of epochs necessary to enable early stopping and stop training earlier remained between 22 and 27 epochs, a similar interval to that found in the digit CNN.Table 2 condenses the letter CNN optimization procedure with the evolution of precision and loss metrics per structure.

TABLE II. EVALUATION OF THE CNN OF LETTERS THROUGH STRUCTURES

Structure	Precision	Loss
Structure 1	98.10%	0.068
Structure 2	98.38%	0.067

Table 3 shows the comparison of the model with other models.

TABLE III. COMPARISON OF DIFFERENT METHODS

Model	Logistic Regression	ANN	SVM	Decision tree	CNN
Precision	0.9510	0.9526	0.9842	0.9921	0.9975
Recall	0.9502	0.9564	0.9815	0.8923	0.9924
F1	0.9509	0.9531	0.8864	0.8964	0.9964
Accuracy	0.9517	0.9561	0.9664	0.9721	0.9921

CNN achieves the highest accuracy, indicating it makes the fewest overall errors across all predictions.

V. CONCLUSION

The work described aims to streamline the correction process of response cards using character recognition. Here's a condensed version:

The work focuses on developing a character recognition method to streamline the correction process of response cards from the Exam Reader software. This involves creating a flow for analyzing response cards, identifying written letters, and automatically correcting responses.

To isolate and identify answers on the cards, image segmentation techniques utilizing fixed marks on the cards are employed. Three convolutional neural networks (CNNs) are trained: one for recognizing digits, one for letters, and one for true or false responses. The achieved accuracies are impressive: 98.84% for digit recognition, 98.38% for letter recognition, and 99.89% for true or false recognition. To integrate the trained CNNs into the Exam Reader application, the weights of the networks are exported and inserted into the application, which is developed using React Native and supports TensorFlow libraries. The application receives the CNN training weights for digit recognition to identify the student's registration number and test code. Upon validation, this information along with the card image is sent to the back-end of the Exam Reader system, where a dedicated service predicts response card images.

REFERENCES

[1]. B. Z. Aljunaeidia, M. S. Alkhasawneh, and M. A. BaniYounes, "Isolated Arabic hand written letters recognition based on contour matching and neural network," *Journal of Computer Science*, vol. 14, no. 11, pp. 1565–1576, 2018.

[2]. M. Das, M. Panda, and S. Dash, "4 A comparative analysis of machine learning techniques for Odia character recognition," *Machine Learning Applications*, pp. 65–90, Apr. 2020.

[3]. M. Das and M. Panda, "An ensemble method of feature selection and classification of Odia characters," in *Proceedings of the 1st Odisha International Conference on Electrical Power Engineering, Communication and Computing Technology, ODICON 2021*, Bhuba, Odisha, January 2021.

[4]. M. Husnain, M. M. Saad Missen, S. Mumtaz et al., "Recognition of Urdu handwritten characters using convolutional neural network," *Applied Sciences*, vol. 9, no. 13, p. 2758, 2019.

[5]. D. Khedidja and M. Hayet, "Multiple classifiers and invariant features extraction for digit recognition," *International Journal of Computer and Electrical Engineering*, vol. 11, no. 1, pp. 41–52, 2019.

[6]. M. B. Bora, D. Daimary, K. Amitab, and D. Kandar, "Handwritten character recognition from images using CNN-ecoc," *Procedia Computer Science*, vol. 167, pp. 2403–2409, 2020.

[7]. R. B. S. D. S. Jana, "Handwritten digit recognition using convolutional neural networks," in *Deep Learning, Bhattacharyya Siddhartha, Snasel Vaclav, Hassanien Aboul Ella, Saha Satadal*, B. K. Tripathy, Ed., pp. 51–68, De Gruyter, Berlin, Germany, 2020.

[8]. M. A. Chhajro, "Handwritten Urdu character recognition via images using different machine learning and deep learning techniques," *Indian Journal of Science and Technology*, vol. 13, no. 17, pp. 1746–1754, May 2020.

[9]. G. Jha and H. Cecotti, "Data augmentation for handwritten digit recognition using generative adversarial networks," *Multimedia Tools and Applications*, vol. 79, no. 47–48, pp. 35055–35068, 2020.

[10]. M. R. Kibria, A. Ahmed, Z. Firdawsi, and M. A. Yousuf, "Bangla compound character recognition using support vector machine (SVM) on advanced feature sets," in *Proceedings of the 2020 IEEE Region 10 Symposium, TENSYMP 2020*, pp. 965–968, Dhaka, Bangladesh, June 2020.

[11]. P. Chakraborty, A. Islam, M. Abu Yousuf, R. Agarwal, and T. Choudhury, "Bangla handwritten character recognition using convolutional neural network," *Machine Intelligence and Data Science Applications*, vol. 132, pp. 721–731, 2022.

[12]. R. C. Sahoo, S. K. Pradhan, and P. Tanwar, "HopNet based associative memory as FC layer in CNN for Odia character classification," in Proceedings of the Confluence 2020 - 10th International Conference on Cloud Computing, Data Science and Engineering, pp. 178–182, Noida, India, January 2020.

[13]. T. Sun, C. Wang, and S. Y. Kim, "Psychometric properties of an English Writing Self-Efficacy scale: aspects of construct validity," Reading and Writing, vol. 35, no. 3, pp. 743–766, 2021.

[14]. V. Makarenkov, L. Rokach, and B. Shapira, "Choosing the right word: using bidirectional LSTM tagger for writing support systems," *Engineering Applications of Artificial Intelligence*, vol. 84, no. SEP, pp. 1–10, 2019.

[15]. T. Sun, C. Wang, and S. Y. Kim, "Psychometric properties of an English Writing Self-Efficacy scale: aspects of construct validity," *Reading and Writing*, vol. 35, no. 3, pp. 743–766, 2021.