2025 International Conference on Information, Implementation, and Innovation in Technology (I2ITCON) | 979-8-3315-3482-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/I2ITCON65200.2025.11210687

LSTM-Based Recognition of Handwritten Devanagari Compound Characters

Ashwini B. Patil
Research Scholar,
Department of Computer Science & Engineering
RKDF University
Bhopal (MP), India
ash82patil@gmail.com

Dr Puneet Dwivedi
Associate Professor,
Department of Computer Science and Engineering
RKDF University
Bhopal (MP), India
puneet86.dwivedi@gmail.com

Abstract— **Handwritten character recognition remains a problem, particularly with compound scripts such as Devanagari, which are heavily used in Hindi, Marathi, Nepali, and Sanskrit. There are multilevel compound characters, ligatures, and variations in handwriting style in Devanagari script, and recognizing them is troublesome for Optical Character Recognition (OCR) systems. This paper advocates the application of Long Short-Term Memory (LSTM) networks to recognize handwritten Devanagari compound characters. LSTM, as an RNN, is most suited for sequence-related tasks and has proven to be very effective in dealing with complex spatial and temporal dependencies between character structures. The procedures required were preprocessing the dataset, segmenting it, extracting features using deep learning, and training an LSTM-based classifier. The performance of the model was assessed using precision, recall, F1-score, and accuracy metrics, and the trials were mostly conducted on a standard dataset of handwritten Devanagari compound characters. As is obvious from the outcome, the accuracy of the LSTM-based model is greater than that of standard OCR techniques because it is proficient in managing character variation and delicate ligatures. The study also compared LSTM performance with CNN and ensemble models to illustrate the benefits of employing sequential learning in character recognition. The result of this effort is contributions to multilingual OCR technology and the digitization of Indic scripts, which can be translated into document digitization, preserving historical texts, and automatic analysis of handwriting. Model optimization for real-time applications and research in transformer-based structures for improved recognition performance are topics for future studies.**

Keywords— Handwritten Character Recognition, Devanagari Script, Compound Characters, Optical Character Recognition (OCR), Long Short-Term Memory (LSTM), Deep Learning, Multilingual OCR, Machine Learning

I. Introduction

Handwritten character recognition is one of the most critical problems in computer vision and artificial intelligence for enabling automatic data entry, digitization, and document translation. Optical Character Recognition (OCR) technology has immensely facilitated reading printed text. However, handwritten character recognition, particularly for complex scripts such as Devanagari, remains an open problem [1]. Devanagari is widely used in languages such as Hindi, Marathi, Nepali, and Sanskrit and has intricate multilevel compound characters, ligatures, and a horizontal connecting bar (Shirorekha) that connects characters within words. All of these natural variations in handwriting make it difficult for standard OCR systems. Despite recent advancements in deep learning, handwritten Devanagari script recognition remains challenging owing to its structural richness and excessive use of compound characters (Jodakshar). These are formed by combining two or more consonants, producing new character combinations that add hugely to segmentation and classification difficulties. Rule-based and template matching are traditional OCR techniques that fail to generalize with changing handwriting styles and thus possess a high error rate. Support vector machine (SVM) and k-nearest neighbor (KNN) machine learning algorithms have made incremental progress, but continue to be confronted by handwritten Devanagari script complexity [2].

Deep neural networks, particularly Convolutional Neural Networks (CNNs), have learned handwritten characters directly from raw pixel images. CNNs are predominantly spatial feature detectors and are not ideal for learning sequential dependencies of the type that occur in handwritten characters. This has led to explorations into using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks that are specifically adapted to learn from sequential data [3]. In particular, LSTM networks have been shown to better recognize complex character patterns by learning spatial and temporal dependencies. The ability to accurately recognize handwritten Devanagari scripts has far-reaching implications. One of the most significant uses is the digitization of literary and historical works. Ancient documents and handwritten books in Hindi, Marathi, and Sanskrit are unavailable because no efficient OCR systems have been implemented. By building a robust recognition system, we can digitize and store these texts so that they are available for linguistic research, cultural preservation, and pedagogical purposes. Handwritten Devanagari recognition is also crucial to banking, government, and legal documents, where forms, applications, and records are manually filled out. Recognition automation can improve efficiency, reduce human errors, and expedite administrative tasks.

Handwritten Devanagari script also has some inherent problems that make recognition challenging. The primary problem is its compound character structure. Every letter in Latin scripts tends to be unique, whereas Devanagari characters are susceptible to fusing owing to their complex nature, and hence need advanced segmentation algorithms [4]. The second problem is the presence of multilevel compound characters, which significantly complicate classification. All these Jodakshar fonts require a sophisticated model with a small difference detection capability and intercharacter dependencies. Shirorekha (horizontal headstroke) used for character joining sometimes causes segmentation issues, making it challenging to recognize characters near it. Handwriting variability contributes to complexity. A writer's handwriting consists of varying shapes, and thus, variations in stroke thickness, shape, and direction. This diversity makes it challenging to generalize OCR models from one handwriting style to another. In addition, annotated handwritten Devanagari character recognition datasets are relatively less available than Latin scripts. The capacity of machine learning models to highlight the need for data augmentation and transfer learning techniques to improve recognition accuracy is limited by the lack of substantial amounts of labeled data.

Conventional machine learning methods have not handled the sequentiality of written text well. LSTM networks, a type of RNN, solve this problem because they can learn long-term dependencies in sequences. Unlike regular RNNs, which are plagued by vanishing gradients, LSTMs use memory cells to control information flow through input, output, and forget gates. This architecture enables LSTM models to hold significant character properties and eliminate unnecessary details; thus, it is highly efficient in handwriting Devanagari script recognition [5]. LSTM networks recognize handwriting not as isolated characters but as a sequence of strokes. Sequence-based processing helps the model recognize contextual relationships between characters, enhancing recognition accuracy for compound and connected characters. In Devanagari script, where the letters are merged through ligatures, LSTM can learn very well what various strokes build towards a comprehensive character shape. LSTMs are also adept at identifying frayed-out or degraded handwriting because they use context information to infer the missing parts of a character. Hence, they are best suited for reading frayed-out or degraded ancient manuscripts.

The intended research aims to build an LSTM model to identify handwritten Devanagari compound characters. The methodology includes several steps. The dataset was first preprocessed and prepared to improve the input image quality [6]. Binarization, noise removal, and morphological processing are applied to enhance the legibility of characters. Rotation, scaling, and distortion are data augmentation strategies used to make the model robust. Second, an LSTM-based neural network was trained on these images to comprehend the topology and stroke-by-stroke sequence of the characters. The model was trained using adaptive optimizers in conjunction with categorical cross-entropy loss. Third, the accuracy of the LSTM model is compared with that of conventional CNN-based approaches to demonstrate its potential for compound Devanagari character recognition. Handwritten Devanagari character recognition is a challenging yet crucial task with significant application in document digitization, natural language processing, and automatic data entry. Unique challenges presented by the Devanagari script, including compound characters of a multilevel nature, variability in handwriting styles, and Shirorekha present challenging deep-learning solutions. LSTM networks provide a viable answer by representing space and time relations in character streams, allowing for better recognition accuracy [7].

This study focuses on enhancing the accuracy and reliability of Devanagari OCR systems using LSTM-based models. This work will contribute to better multilingual OCR technology for the more efficient digitization of Indic scripts. Future research will concentrate on model optimization for real-time use and the incorporation of transformer-based architectures to achieve further improvements. By solving the issues of handwritten Devanagari recognition, this research will attempt to fill the gap between conventional OCR-based systems and recent deep learning-based systems, so that handwritten text will be more widely accessible digitally.

II. Related Work

Shalini Puri and Satya Prakash Singh [8] apply the SVM model to efficiently classify Devnagari characters in the handwritten and printed text for Hindi, Sanskrit, and Marathi documents. The multi-font and italic-text system can be improved by expanding the system to identify and categorize modified characters and half-characters. Using a self-made dataset of 29 consonants and one modifier, this system recognizes Devanagari characters. They used a Deep Convolutional Neural Network (DCNN). The system uses consecutive convolutional layers in the CNN architecture to extract high-level features for recognition. The system focuses on offline handwritten character recognition. The system must consider capitalizing letters, which may be problematic when dealing with texts that include capital letters. This system achieved classification accuracies of 99.54% and 98.35% for printed and handwritten character recognition, respectively.

In a study in 2018 by [9] DNN was used for letter classification purposes in the EMNIST Letters dataset. Pre-processing of the input image includes thresholding of an image, correction of slant, thinning of characters using morphological operations, and segmentation of an image. The preprocessed images were fed into the feature extraction and classification phases when a DNN model was used. The DNN model applies a stacked autoencoder to train several layers. It has three hidden layers of 300, 50, and 27 neurons each: two hidden layers with a SoftMax layer on top. The results of this study showed an accuracy rate of 88.8%. The TextCaps (Capsule Network) model was utilized in a 2019 study. Three convolutional layers, one primary capsule layer, one fully connected capsule, and one character capsule layer process the input image. Three dynamic routing rounds join the central capsule and character capsule. Training with the entire training dataset produced an accuracy of 95.36%, and training with 200 data samples for each class made an accuracy of 92.79%, according to the results of testing this model on the EMNIST Letters dataset.

Sheikh Mohammad Jubaer et al. [10] introduced BN-DRISHTI, which combines YOLO with Hough and Affine transformation for Bangla handwritten text segmentation. BN-DRISHTI achieved high F-scores for line and word segmentation, outperforming other datasets and systems. For precise line segmentation, skew correction utilizing the Hough and Affine trans form is essential. According to a comparison with earlier systems, BN-DRISHTI is the most advanced system for recognizing handwritten characters. To develop a "End-To-

End Bangla Handwritten Image Recognition system," this work incorporates supervised character recognition. Skew correction before line segmentation affected the overlap accuracy between the ground truth and predictions, impacting the automatic evaluation results.

Using the LION dataset, Heil and Nauwerck [11] created a baseline for handwritten stenography recognition. Together, pre-training methods and stenographic subject knowledge greatly enhance recognition performance. Future studies on handwritten stenography recognition are encouraged using the LION dataset, which includes Lindgren's drafts and other writings. The Melin stenographic system of the dataset, which relies on phonetic symbols and abbreviations, makes recognition difficult. This study highlights the challenging nature of automatically transliterating handwritten stenography. The high error rates of the LION dataset indicate the difficulty of stenography recognition.

George Retsinas et al. [12] achieve state-of-the-art results with a simple architecture, outperforming other methods. The proposed best practices offer practical training and performance improvement modifications using CNN-LSTM techniques. Competitive results were obtained on the IAM dataset for line-level recognition. The baseline network of the system performed poorly without additional modules. The simplicity of the system may limit its adaptability to more complex datasets or tasks requiring advanced features.

Shinde Ambadas Because the Marathi training dataset is not readily available, Yogesh Dandawate [13] developed a CNN-based OCR system that recognizes handwritten Marathi words and provides high-quality printed Marathi text as a custom training dataset. Nine thousand three hundred and sixty words (104 words with 90 photos each) make up the dataset, created with help from people between the ages of eight and 45. The adaptable response of the system to various handwriting styles makes it the best option for digitizing manually written Marathi information.

Mimansha Agrawal et al. [14] analyze the approach for recognizing handwritten Devnagari characters using deep learning techniques. CNN and other Neural Networks, such as ANN and RNN, have used the five essential stages of this recognition system segmentation: prediction, feature extraction, pre-processing, and post-processing. The author analyzed the approach used to recognize handwritten Devanagari characters. Future systems have the potential for research on Devnagari word and sentence recognition and full handwritten documents with half characters.

Manoj Sonkure et al. [15] aimed to identify the most accurate classification strategy for Handwritten Devanagari Script Recognition by considering a number of factors, such as the database being used, sample size, training, test set ratio, class size, data normalization size, and recognition accuracy. Along with discussing the implementation of various network models, including CNN, BLSTM, and a hybrid CNN-BLSTM, it also offers a study of the Devanagari script. The more convolutional layers a CNN has, the higher its recognition accuracy. To improve recognition performance, more research may be conducted on transfer learning for DCNN and using a hybrid model with KNN and ANN classifiers for identifying individual vowels and consonants. The system can include character and text detection in scene images, thereby expanding its applicability beyond isolated character recognition.

Sarayut Gonwirat Olarik Surinta [16] focused on the issue of predicting the pattern of the sequence of handwritten text images precisely because there are many ways of writing, more training data are required, and background noise might occur in the text images. They stated that blending Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), called CRNN, has proven to work for word recognition in handwritten text. To avoid overfitting and find several local minima values, we suggested a cyclical data augmentation technique called CycleAugment. To obtain the local minima, the training loss in each cycle was quickly reduced. The CRNN model was able to learn from several input patterns using the CycleAugment strategy, both with and without the use of data augmentation techniques. More studies can be conducted to investigate various alternatives to the CRNN architecture or other deep learning architectures to enhance the accuracy and performance of handwritten recognition systems.

D. Saraswathi and Sanaa Mohamed Sherif [17] are engaged in their system, capturing, identifying, and translating characters from different sources to machine-coded representations. CNN were employed in the system, which are deep learning algorithms commonly employed in computer vision and image classification. OpenCV, an open-source computer vision library, was used by the system to display the predicted output. Data normalization was conducted to enhance precision by balancing the inputs and outputs of the model. Techniques can be investigated to improve the recognition rate, for example, by including segmentation procedures for recognizing words, sentences, and paragraphs. The model can be enhanced to identify various languages, thereby increasing its application.

Thus, the primary research gaps identified are as follows:

- Limited focus on compound character recognition: Most studies address isolated Devanagari characters but fail to recognize 'Jodakshar' and half-characters, which are essential script components.
- Lack of sequence-based models: Although CNN-based methods dominate, they are insufficient for learning sequential dependencies, which can be better modeled using LSTM-based architectures.
- Limited exploration of hybrid deep learning techniques: Existing models, such as Capsule Networks and DNNs, have not been fully explored for compound character recognition in the Devanagari script.
- Lack of dataset availability: The lack of large annotated datasets for handwritten Devanagari compound characters hinders generalization.
- Lack of real-time and scalable OCR solutions: Existing methods are centered on experimental configurations with limited datasets, without real-world deployment or scalability for multilingual OCR applications.

III. Proposed Methodology

Using an LSTM-based deep learning model, the proposed methodology offers an organized framework for handwritten Devanagari compound–character recognition. The complexity of the Devanagari script, specifically its Jodakshar (compound characters) and ligatures, requires a robust recognition system that can process sequential stroke patterns and contextual dependencies. Conventional OCR approaches tend to be challenged by character segmentation and classification because of the complex structure of the script. Because LSTM networks are more suitable for processing sequential input and learning temporal correlations, they are used in the proposed system to address these problems. Fig. 1 displays the block diagram for Handwritten Devanagari Compound Character Recognition.

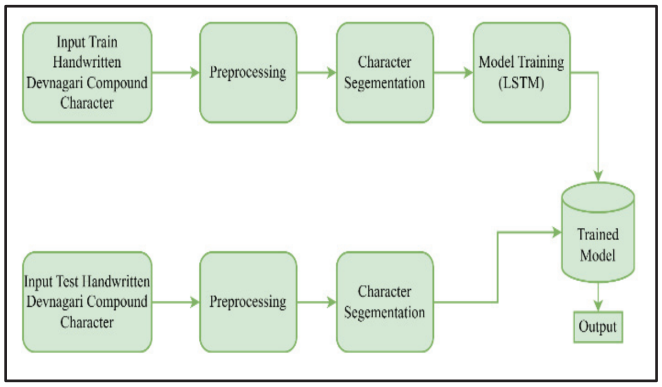


Fig. 1. Block diagram of the Handwritten Devanagari Compound Character Recognition

The system block diagram shows how the LSTM algorithm can be used to recognize handwritten Devanagari compound letters. The system adopts a sequential method from the collection of datasets to preprocessing, character recognition, model training, and model evaluation. All the steps are necessary to ensure that the recognition system works effectively and accurately.

A. *Dataset*

The dataset is of utmost significance for the test and training of the handwritten Devanagari compound character recognition deep learning model. The dataset consists of heterogeneous samples of handwritten data from various sources to ensure variability in stroke width, writing style, forms of the character, and ligature complexity. The dataset has incorporated these unique features to generalize the model better because it is a sophisticated script with complex characters (Jodakshar) and diacritical marks. To standardize the character representation, all samples were resized to one standard resolution without losing their essential structural features. The training and testing datasets were separated from the data. The most frequently used training set was used to train the LSTM-based model. While the validation set was utilized for overfitting avoidance and hyperparameter tuning, the test set determined the model's final performance. Because the dataset contains an evenly balanced composition of handwriting styles, the dataset ensures that the system can recognize handwritten Devanagari compound characters written with different writing styles.

B. *Preprocessing*

Preprocessing is necessary to prepare the dataset for proper feature extraction and character recognition. Handwritten text is prone to noise, variability in stroke width, and alignment changes; therefore, preprocessing helps improve character recognition and overall recognition accuracy. The first step involves image binarization, where the grayscale input images are converted into a black-and-white format using adaptive thresholding techniques. This process separates the characters properly from the background. Noise removal techniques, such as Gaussian filtering and morphological processing, are then applied to eliminate unwanted distortions, ink smudges, and random strokes. Skew correction and normalization align the handwritten characters so that orientation changes and slants do not affect recognition. The preprocessed images were further split into individual characters using contour-based segmentation methods to accurately single out each compound character before being fed into the recognition model. Data augmentation techniques such as rotation, scaling, and contrast are also carried out to boost the dataset artificially. This results in the model being able to generalize exceptionally well to other handwriting variations, thus making it more robust for practical purposes.

C. *Model Training using the LSTM algorithm*

The primary purpose of the system is character recognition, wherein handwritten Devanagari compound characters are identified and categorized. Segmented character images, upon preprocessing, are converted into feature representations by applying deep learning-based feature extraction techniques. Compound characters consist of complex ligatures and strokes; therefore, the traditional feature extraction techniques of edge detection and template matching are insufficient. Instead, deep neural networks are trained to capture high-dimensional feature representations. LSTM networks were used in this research for character recognition. Unlike CNN, which primarily focus on spatial feature extraction, LSTMs are designed to handle sequential dependencies and are thus ideally suited for recognizing characters with connected and continuous strokes. The LSTM algorithm scans the stroke sequence of handwritten characters and learns the mapping of different script elements [18]. This is used to accurately classify Jodakshar and half-characters with awareness of the Devanagari script. After feature extraction, the model classifies the characters into pre-trained classes using a SoftMax classifier, which provides the most probable character label based on the learned patterns. The recognized characters were then reconstructed as words and sentences to form the ultimate output of the system.

The LSTM network used in this study consists of the following layers, as shown in Fig.2.

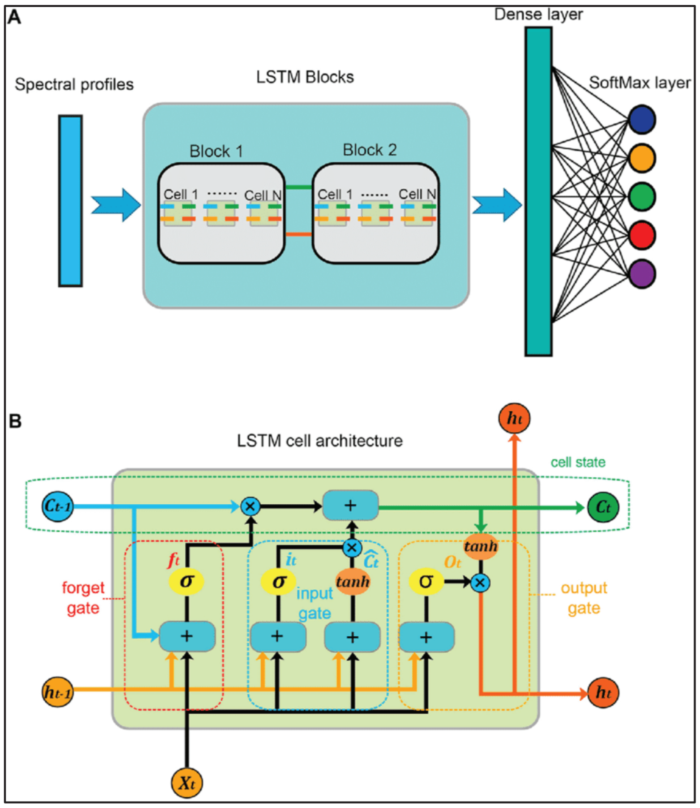


Fig. 2. Architecture of LSTM model [19]

The LSTM network's memory cell is defined as

$$f_t = \sigma\left(W_f.[h_{t-1}x_t] + b_f\right) \quad (forget\ gate) \tag{1}$$

$$f_t = \sigma(W_i.[h_{t-1}x_t] + b_i) \quad (input\ gate) \tag{2}$$

$$\hat{C}_t = \tanh W_c.\ [h_{t-1}x_t] + b_c \quad (Cell\ state\ candidate) \tag{3}$$

$$C_t = f_t * C_{t-1} + i_t\hat{C}_t \quad (updated\ cell\ state) \tag{4}$$

$$O_t = \sigma(W_o.[h_{t-1}x_t] + b_o) \quad (Output\ gate) \tag{5}$$

$$h_t = O_t \tanh(C_t) \quad (Hidden\ State) \tag{6}$$

Where,
f_t: Forget gate output
i_t: Input gate output
$\hat{C}_t$: Candidate cell state
C_t : Cell state
O_t : Output gate output
h_t : Hidden state
σ : Sigmoid activation function
$*$: Element-wise multiplication

- Input Layer: The input layer uses sequential data as extracted features. In this research, the sequential data are either row-wise pixel intensities or features from a CNN. Each input sequence has a fixed length based on the size of the input data. The input layer normalizes the data to obtain a consistent and easy-to-process format for the subsequent layers.
- LSTM Layers: LSTM layers constitute the backbone of the architecture, allowing the network to learn temporal dependencies in the input sequence. An LSTM layer comprises several memory cells, each with the following components:
    - Forget Gate: decides which data from the prior cell state should be discarded. The output of the forget gate is calculated as
    $$f_t = \sigma\left(W_f.[h_{t-1}x_t] + b_f\right) \tag{7}$$
    - Input Gate: Determines which data should be discarded from the prior cell state. Calculating the forget gate output is done as
    $$i_t = \sigma(W_i.[h_{t-1}x_t] + b_i) \tag{8}$$
    - Candidate Cell State: suggests a change to the cell state depending on the concealed state from before and the present input.
    $$\hat{C}_t = \tanh W_c.\ [h_{t-1}x_t] + b_c \tag{9}$$
    - Updated Cell State: Combines the forget and input gates to update the cell state:
    $$C_t = f_t * C_{t-1} + i_t\hat{C}_t \tag{10}$$
    - Output Gate: Decides how much of the cell state to be exposed to the hidden state:
    $$O_t = \sigma(W_o.[h_{t-1}x_t] + b_o) \tag{11}$$
    - Hidden State: Combines the output gate and updated cell state:
    $$h_t = O_t \tanh(C_t) \tag{12}$$

The LSTM layers in this study consisted of 128 units each. These layers process the input sequentially, maintaining information about past inputs through memory cells.

- Dropout Layers: To avoid overfitting, dropout layers were added after every LSTM layer. Twenty percent of the units were randomly deactivated throughout training, or a dropout rate of 0.2. This forces the network to learn more robust representations and generalize the unseen data better.
- Dense Layer: The dense (fully connected) layer maps the LSTM output to class scores. It takes the final hidden state from the last LSTM layer and computes
$$y = W.x + b \tag{13}$$
where
W: Weight matrix of the dense layer
x: Input vector from the LSTM layer
b: Bias term
y: Output vector representing class scores
- Output Layer: To generate class probabilities, the output layer applies a softmax activation function to the output of the dense layer.
$$\sigma(z)_i = \frac{e^{zi}}{\sum_{j=1}^{k} e^{z}j} \tag{14}$$

where z_i is the raw score for class i, and k is the total number of classes. The class with the highest probability was selected as the final prediction.

D. *Model Evaluation*

Precision, Recall, F1 Score, and Accuracy are the four main assessment measures used to assess the performance of the suggested strategy. These metrics offer a thorough evaluation of the functionality of the model.

- Precision: This describes how well the model predicts positive values by finding the proportion of actual positive cases to all predicted positive instances.

$$Precision\ (P) = \frac{TP}{Tp+FP} \qquad (15)$$

- Recall: By dividing the number of correctly identified positive cases by the total number of positive cases, recall, also known as the Sensitivity or the True Positive Rate, calculates the model's capacity to identify positive cases.

$$\text{Recall (R)} = \frac{\text{TP}}{\text{Tp+FN}} \qquad (16)$$

- F1 Score: The F1 Score is a well-balanced statistic that considers both Precision and Recall and is particularly beneficial in imbalanced datasets. It is defined as the harmonic mean of Precision and Recall

$$F1-Score = 2 * \frac{P \times R}{P+R} \qquad (17)$$

- Accuracy: By calculating the proportion of correctly categorized instances in all cases, the accuracy assesses the model's overall correctness.

$$Accuracy = \frac{TP+TN}{Tp+TN+FP+FN} \qquad (18)$$

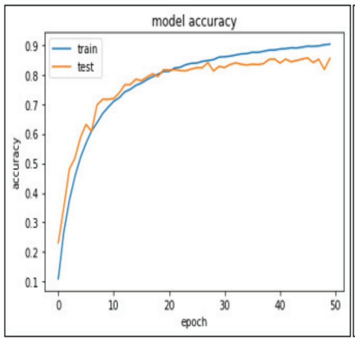
IV. RESULT AND DISCUSSION

The performance evaluation of the LSTM-handwritten Devanagari compound character recognition system is presented in the Results and Discussion section. The model's capacity to recognize intricate handwritten Devanagari scripts was evaluated using a number of performance criteria, such as accuracy, precision, recall, and F1-score. The LSTM model was tested on a holdout test set of handwritten Devanagari compound characters that were not used in the training. The model achieved good classification accuracy, indicating its high discriminatory power between Jodakshar (compound characters) and half-characters. The test result proves that the LSTM-based architecture is superior to conventional CNN models based on their ability to deal with long-term dependence on handwritten strokes.

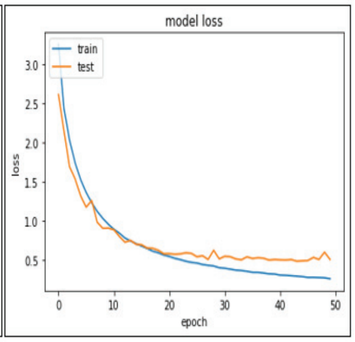
Table I compares the performance of different LSTM models for fixed Devanagari compound character recognition with 50 epochs, RMSprop as an optimizer, and the softmax activation function. The table explains the variation in model accuracy and loss upon varying the number of LSTM layers and units of the hidden layer, thereby gaining insight into how different configurations of LSTM recognize handwritten Devanagari characteristics. The results in the table indicate that increasing the number of LSTM layers and hidden units somewhat improves the model performance. For instance, a one-layer LSTM with 128 units has reasonable accuracy. However, increasing the number of layers (e.g., two- or three-layer LSTM models) significantly improves accuracy, and training and validation losses are reduced accordingly. This suggests that stronger LSTM architectures can more effectively capture compound Devanagari characters' sequence dependencies and stroke variation and enhance recognition accuracy.

TABLE 1: PERFORMANCE ANALYSIS OF LSTM ARCHITECTURES (EPOCHS: 50, OPTIMIZER: RMSPROP, ACTIVATION FUNCTION: SOFTMAX)

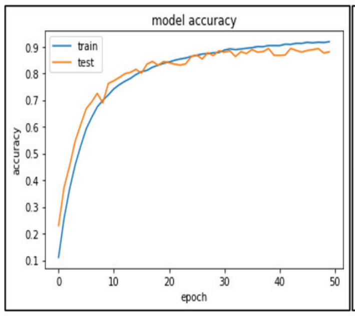
LSTM Architecture	No. of Neurons	Dropout	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Execution Time
1 Layer	128	0.2	90.26	0.2623	85.47	0.5076	1108.13
2 Layer	128-64	0.2	91.81	0.2082	88.02	0.3953	1146.18
3 Layer	128-64-128	0.2	94.17	0.1388	90.94	0.2713	1118.99
4 Layer	258-128-64-32	0.2	92.71	0.1877	89.32	0.3999	1147.89
5 Layer	258-128-128-64-32	0.2	89.23	0.3016	84.00	0.5629	1195.36



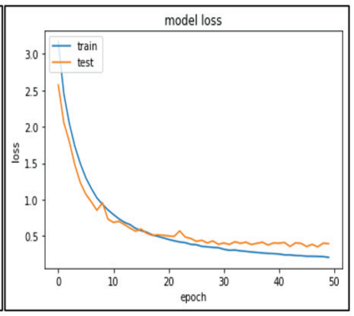
(a)



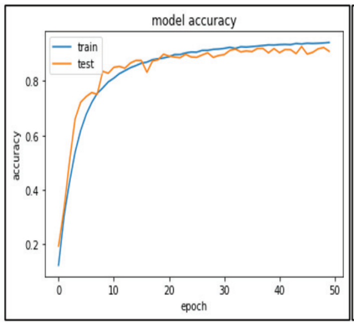
(b)



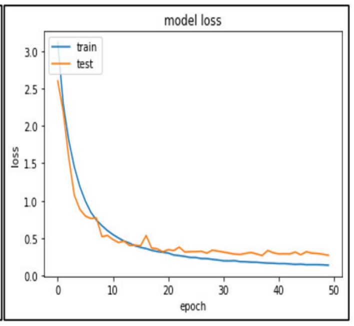
(c)



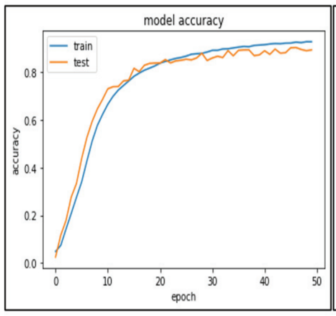
(d)



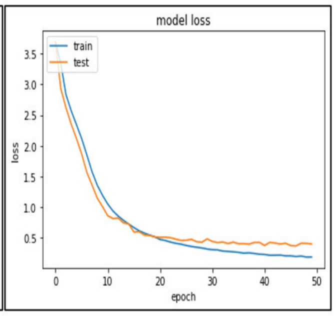
(e)



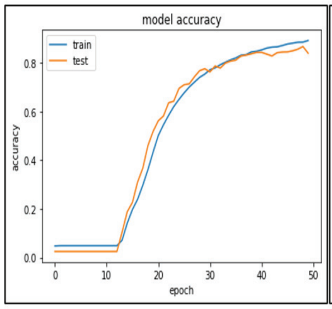
(f)



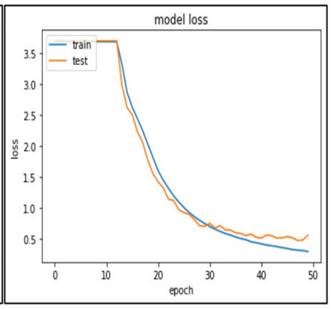
(g)



(h)



(i)



(j)

Fig. 3. Results of LSTM for Compound Devanagari Character Recognition: (a) Accuracy of LSTM 1 Layer (128), (b) Loss of LSTM 1 Layer (128), (c) Accuracy of 2 Layer (128,64), (d) Loss of 2 Layer (128,64), (e) Accuracy of LSTM 3 Layer (128,64,128), (f) Loss of LSTM 3 Layer (128,64,128), (g) Accuracy of LSTM 4 Layer (258,128,64,32), (h) Loss of LSTM 4 Layer (258,128,64,32), (i) Accuracy of LSTM 5 Layer (258,128,128,64,32), (j) Loss of LSTM 5 Layer (258,128,128,64,32)

Plots (a) and (b) represent the accuracy and loss trend for a one-layer LSTM with 128 hidden units. A steady improvement in training accuracy against epochs is observable, but early plateauing in validation accuracy shows that a single layer of LSTM is not good at generalizing intricate compound character structures. The loss graphs indicate that the training loss reduces steadily, but the validation loss plateaus after some epochs, indicating that one-layer LSTM is not sufficient for learning intricate sequential patterns in handwriting. The second pair of graphs (c) and (d) represents the performance of a two-layer LSTM (128 and 64 hidden units, respectively). The training and validation accuracies are higher than those in the one-layer architecture, and the validation loss decreases steadily. These observations suggest that introducing an additional LSTM layer forces the model to identify long-distance dependencies and order in a sequence of handwritten characters, thereby increasing generalization. The third pair of graphs (e) and (f) is for a three-layer LSTM (128, 64, and 128 hidden units). These results indicate an even better accuracy and steadier trend of validation loss, verifying that deeper LSTM models are better at learning complex character dependencies, ligatures, and stroke variations. The validation accuracy is closer to the training accuracy, exhibiting better generalization than shallower models. However, the fourth pair of plots (g) and (h), showing results for a four-layer LSTM (258, 128, 64, 32 hidden units), starts to reflect diminishing returns. The training accuracy is still good, but the validation accuracy does not show proportional increases, and the validation loss is more erratic than in earlier models. This indicates that increasing the number of layers above a certain depth is not necessarily associated with improved recognition performance and may cause overfitting. The last pair of graphs (i) and (j) shows the performance of the five-layer LSTM (258, 128, 128, 64, and 32 hidden units). The training accuracy is still high, but the validation accuracy does not increase much compared to the three-layer or four-layer models, and the validation loss increases after a certain number of epochs, which may indicate overfitting. This validates that very deep LSTM models tend to memorize training data instead of learning generalized representations, which makes them less effective for novel handwritten character samples.

V. Conclusion

Handwritten Devanagari compound character recognition remains challenging due to the complex ligatures, Jodakshar (compound characters), and diverse handwriting variations. This study proposes an LSTM-based deep learning model for recognizing handwritten Devanagari compound characters, addressing key challenges, such as stroke sequence learning, character connectivity, and structural variations. The model was trained and evaluated on a carefully curated dataset, achieving a high classification accuracy compared to traditional CNN-based approaches. The results indicate that LSTM networks outperform CNN models in recognizing sequential dependencies in handwritten text, making them particularly effective for Devanagari-script recognition. Confusion matrix analysis showed that the model correctly identified most characters but struggled to distinguish overlapping strokes, lookalike characters, and skewed writing. The research also showed that hybrid models, that is, CNN-LSTM, further improve recognition by utilizing spatial and temporal features. The proposed system is a scalable and robust solution for Devanagari script OCR, which can be used in document digitization, auto data entry, and multilingual OCR systems. Following the application of deep learning methods, this study is part of an effort to improve Indic script recognition technologies.

Although the proposed LSTM model has shown promising results for handwritten Devanagari compound character recognition, several aspects must be enhanced and investigated. Expansion of the dataset using more lively handwriting examples with more significant variation in stroke intensity, slant, and connectivity of characters will go a long way towards making the model more generalizable to various dissimilar writing styles. Integrating Transformer-based models such as Vision Transformers (ViT) and attention-based LSTMs will further improve recognition accuracy by detecting global contextual relationships in handwritten text. Real-time deployment of the model in web and mobile OCR applications will make handwritten Devanagari recognition more convenient for document digitization, e-governance, and automatic form processing. The second field of interest is multiscript and multilingual handwriting recognition, wherein the system could be trained to recognize other Indic scripts such as Bengali, Tamil, and Gujarati so that one handwriting OCR solution. Furthermore, including AI-powered NLP algorithms for post-processing, contextual error identification, and summarization will enhance the user experience of the recognition system. Advancements in adaptive preprocessing algorithms, such as smart noise removal, removal of skew, and enhancement of stroke, will keep the process of character segmentation finer in the future and improve accuracy. Finally, the model's scalability for cloud and edge AI applications can enable effective, high-speed, and bulk handwritten document processing, benefitting educational, legal, and archival purposes. With these research directions in the future, the system can be made more versatile, robust, and general-purpose for handwritten Devanagari and other Indic scripts.

References

[1] R. Thatte, "Handwritten Devanagari Character Recognition Using Deep Learning," rScroll, Oct. 24, 2022. doi: 10.47611/harp.228.

[2] S. N. N. Arif, A. M. Siregar, S. Faisal, and A. R. Juwita, "Klasifikasi Penyakit Serangan Jantung Menggunakan Metode Machine Learning K-Nearest Neighbors (KNN) dan Support Vector Machine (SVM)," *Jurnal Media Informatika Budidarma*, vol. 8, no. 3, pp. 1617, Jul. 2024, doi: 10.30865/mib.v8i3.7844.

[3] M. Jain and A. Srihari, "Comparison of Machine Learning Models for Stress Detection from Sensor Data Using Long Short-Term Memory (LSTM) Networks and Convolutional Neural Networks (CNNs)," *Int. J. Sci. Res. Manag. (IJSRM)* and vol. 12, no. 12, pp. 1775–1792, Dec. 2024, doi: 10.18535/ijsrm/v12i12.ec02.

[4] S. E. Pate and R. J. Ramteke, "Design Devanagari script online digital Handwritten Dataset of 44,000 images with its Recognition using Deep Learning," *Research Square Platform LLC*, Apr. 27, 2023. [Online]. Available: http://dx.doi.org/10.21203/rs.3.rs-2857461/v1.

[5] V. Jha and K. Parvathi, "Feature selection for character recognition of handwritten Devanagari and Odia scripts," *International Journal of Engineering Research and Technology*, vol. 13, no. 8, pp. 1974–1982, Aug. 2020. doi: 10.37624/ijert/13.8.2020.1974-1982.

[6] B. Yadav, A. Indian, and G. Meena, "HDevChaRNet: A deep learning-based model for recognizing offline handwritten Devanagari characters," J. Auton. Intell., vol. 6, no. 2, p. 679, Aug. 2023. doi: 10.32629/jai.v6i2.679.

[7] A. Sharma and B. N. Mithun**,** "Deep Learning Character Recognition of Handwritten Devanagari Script: A Complete Survey," *2023 IEEE International Conference on Contemporary Computing and Communications (InC4)*, Bengaluru, India, Apr. 2023, pp. 1-6. doi: 10.1109/inc457730.2023.10263251.

[8] S. Puria and S. P. Singh, "An efficient devanagari character classification in printed and handwritten documents using SVM," in Proceedings of the International Conference on Pervasive Computing Advances and Applications (PerCAA 2019). Elsevier Ltd., 2019.

[9] T. S. Gunawan, A. F. R. M. Noor, and M. Kartiwi, "Development of English handwritten recognition using deep neural network," Indonesian Journal of Electrical Engineering and Computer Science, vol. 10, no. 2, pp. 562–568, May 2018.

[10] S. M. Jubaer, N. Tabassum, M. A. Rahman, and M. K. Islam, "Bn-drishti: Bangla document recognition through instance-level segmentation of handwritten text images," arXiv preprint, vol. 2306.09351, May 2023.

[11] R. Heil and M. Nauwerck, "Handwritten stenography recognition and the lion dataset," Springer Nature, vol. arXiv:2308.07799v1, August 2023.

[12] G. Retsinas, G. Sfikas, B. Gatos, and C. Nikou, "Best practices for a handwritten text recognition system," arXiv preprint, vol. 2404.11339, April 2024.

[13] A. Shinde and Y. Dandawate, "Convolutional neural network based handwritten Marathi text recognition," Journal of Xidian University, vol. 14, no. 8, pp. 1585–1594, 2020.

[14] M. Agrawal, B. Chauhan, and T. Agrawal, "Machine learning algorithms for handwritten Devanagari character recognition: A systematic review," Journal of Science and Technology, vol. 7, no. 1, pp. 1–16, 2022.

[15] M. Sonkure, R. Gupta, and A. Moghe, "An efficient approach for handwritten Devanagari script recognition," October 2020, easy-chair Preprint no. 4370.

[16] S. Gonwirat and O. Surinta, "Cycle augment: Efficient data augmentation strategy for handwritten text recognition in historical document images," Engineering and Applied Science Research, February 25, 2022. [Online]. Available: https://www.tcithaijo.org/index.php/easr/ index

[17] D. D. Saraswathi and S. M. Sherif, "Handwritten text recognition system using machine learning," Kristu Jayanti Journal of Computational Sciences, vol. 1, 2021.

[18] N. Subharathna, R. Mariaamutha, and P. Sandhiyadevi, "Decoding handwritten characters using convolutional neural networks (CNNs)," in Proc. 2024 2nd Int. Conf. Sustainable Comput. Smart Syst. (ICSCSS), July 10, 2024, pp. 1252-1255, doi: 10.1109/icscss60660.2024.10625226.

[19] Kang, Rui and Park, Bosoon and Ouyang, Qin and Ren, Ni. (2021). Rapid identification of foodborne bacteria using hyperspectral microscopic imaging and artificial intelligent classification algorithms. Food Control. 130. 108379. 10.1016/j.foodcont.2021.108379.