Journal of Autonomous Intelligence (2023) Volume 6 Issue 2
doi: 10.32629/jai.v6i2.679

ORIGINAL RESEARCH ARTICLE

HDevChaRNet: A deep learning-based model for recognizing offline handwritten devanagari characters

Bharati Yadav, Ajay Indian*, Gaurav Meena

Department of Computer Science, Central University of Rajasthan, Ajmer 305817, India

*** Corresponding author:** Ajay Indian, ajay.indian@curaj.ac.in

ABSTRACT

Optical character recognition (OCR) converts text images into machine-readable text. Due to the non-availability of several standard datasets of Devanagari characters, researchers have used many techniques for developing an OCR system with varying recognition rates using their own created datasets. The main objective of our proposed study is to improve the recognition rate by analyzing the effect of using batch normalization (BN) instead of dropout in convolutional neural network (CNN) architecture. So, a CNN-based model HDevChaRNet (Handwritten Devanagari Character Recognition Network) is proposed in this study for same to recognize offline handwritten Devanagari characters using a dataset named Devanagari handwritten character dataset (DHCD). DHCD comprises a total of 46 classes of characters, out of which 36 are consonants, and 10 are numerals. The proposed models based on convolutional neural network (CNN) with BN for recognizing the Devanagari characters showed an improved accuracy of 98.75%, 99.70%, and 99.17% for 36, 10, and 46 classes, respectively.

Keywords: character recognition; DHCD; deep learning; CNN; batch normalization; dropout

ARTICLE INFO

Received: 2 June 2023
Accepted: 19 July 2023
Available online: 15 August 2023

1. Introduction to offline handwritten Devanagari character recognition (OHDCR)

The scientific community has been researching handwritten text recognition (HTR) systems for the past two decades. The conversion process of handwritten text or documents into digital text is termed HTR[1]. Online HTR and offline HTR are the two basic types into which HTR has been divided[2]. The writer's choice of stroke sequence when writing the text is the key distinction between the two. In an online HTR, the recognition process has access to the order, which aids the recognizer in producing better results.

In contrast, an offline HTR has only a scanned copy of the handwritten documents, which presents numerous challenges in correctly extracting text from images[2,3]. There is still a need to develop more robust methods for extracting and recognizing handwritten text on images, even though offline HTR has been discussed for many years. The major causes are variations in handwriting style and unconstrained handwriting.

As mentioned in the study of Puri and Singh[4], the Brahmi script, the mother script of several Indian languages, is where Devanagari started. The writing and reading script Devanagari is widely used in a broad region of India. Devanagari developed and advanced gradually from Brahmi, going through the following stages:

Brahmi, Bharati (Bhagwat Gita), Gupta, Nagari, and Devanagari scripts. Many Indian languages, including Hindi, Konkani, Marathi, Prakrit, Sanskrit, and Sindhi, are written in Devanagari. Additionally, it serves as the additional script for the languages of Punjabi and Kashmiri. Many distinctive characteristics of Devanagari may be seen, such as the fact that there are no capital or small letters and the letters are not spelled out in any particular order; it is written left to right, top to bottom, and read in the order of sequence; the use of a long, continuous horizontal top line (shirorekha) on the characters is a particularly distinctive aspect of Devanagari; and each character's top lines are connected one by one when characters are combined to make a word, resulting in a single, lengthy shirorekha.

Recently, numerous techniques have been developed for offline Devanagari optical character recognition (OCR). The processing of its documents still needs to be improved as it includes shirorekha, vast character sets, complicated conjuncts, characteristic geometric structure of characters, and linguistic complexities (top line)[4]. The Devanagari handwritten character dataset (DHCD) is a new publicly available dataset comprised of character images segmented from documents written by hand that explores the problems associated with Devanagari character recognition[5].

1.1. Introduction to the problem

The recognition rate is not very high for Devanagari characters. Non-availability of a standard dataset for all characters of the Devanagari script; similarly, the dataset used in our proposed work does not comprise vowels and modifier characters.

Existing challenges in the problem: need to improve the recognition rate of Devanagari characters. Prepare a dataset of all characters (consonants, vowels, and numerals in one place) as well as of modifier characters of the Devanagari script.

Contribution proposed study: the DHCD is utilized in the proposed work. A brief study of the existing systems for Devanagari character recognition is specified in related work section. An attempt to improve the recognition rate of existing systems by analyzing the effect of batch normalization (BN) at the different levels of CNN architecture compared to dropout for recognizing the Devanagari characters efficiently. A Devanagari character recognition system based on deep learning is proposed.

1.2. The general framework of character recognition task

As mentioned in the study of Indian and Bhatia[6], the framework of any character recognition task, in general, consists of several phases, which are briefly described in **Figure 1**.

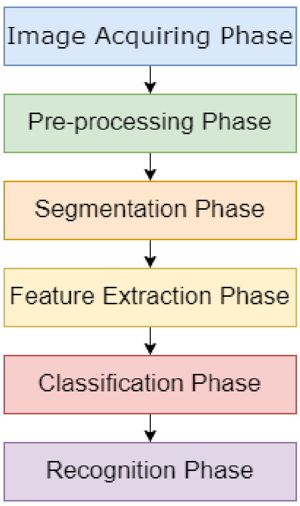


Figure 1. Phases of character recognition framework.

1) Image acquiring phase: this is the phase where a handwritten text or character on paper is turned into a digital image. For this, handwritten text or characters on paper documents are scanned. The next phase is then applied to these digital images.

2) Pre-processing phase: pre-processing aims to remove noise from an image so that the recognition system can work well and give accurate results. The main goal is eliminating noise, normalizing the data, and compressing it without losing important information.

3) Segmentation phase: in this phase, the text or character that has already been processed is broken up into parts, such as paragraphs, sentences, words, and characters. It is a very important phase because being able to separate lines into words and words into characters is directly related to how well one can read handwritten characters. These images can also be turned into binary to be analyzed further.

4) Feature extraction phase: as feature sets are one of the most important parts of a recognition system, a good feature set shows the characteristics of a class in a way that helps it stand out from other classes. The main goal of this phase is to extract the best set of features, which reduces mistakes in recognition and as a result, increases the rate of correct recognition.

5) Classification phase: in this phase, the features extracted in the feature extraction are used to decide which class an input character belongs to. To make a classification model is done using different classification methods, such as CNN, SVM, ANN, KNN, etc.

6) Recognition phase: This is the last phase, and it is responsible for using the classification model made in the classification phase to recognize handwritten characters.

The paper is further organized into six sections. Some of the major studies done on offline character recognition are reviewed in Section 2. Section 3 elaborates on the proposed methodology, covering a brief overview of the dataset along with samples of consonants and numerals from the Devanagari script; an introduction to the convolutional neural network; batch normalization and dropout; and lastly, the proposed model architecture. In Section 4, the proposed models' performances are discussed and compared with other states of the art. Section 5 presents the future direction. At last, Section 6 concludes the present study.

2. Related work

Many researchers with different approaches have attempted offline handwritten character recognition as a task. Much work has been reported to recognize characters written in Indic and non-indic scripts. This section reviews several handwritten character recognition methods that have been used.

Bhalerao et al.[7] achieved an overall recognition accuracy of 95.81% by combining quadratic and SVM classifiers with 3-fold cross-validation. The overall accuracy was computed by averaging the accuracy of each character. A dataset of 29,440 samples collected from different individuals was used for the study.

Singh and Puri[4] proposed an offline Devanagari character classification system utilizing SVM for recognizing the Shirorekha-Less (SL) character from scanned monolingual handwritten and printed Hindi, Marathi, and Sanskrit document images. Features are extracted from SL characters and SL-modified characters. For training, the SVM (gaussian kernel) classifier was employed, then tested using various unidentified scanned text document images, and performance was examined. Both handwritten and printed document images had an average SL classification accuracy of 99.54% and 98.35%, respectively.

For Indian bank cheques, the BCHWTR (bank cheque hand written text recognition) method is proposed by Ghosh et al.[8]. A dataset of 100 individual people's handwritten text on 100 separate bank cheques is created using Latin script. The feature values from the grey level co-occurrence matrix and histogram of oriented gradients were combined to create a final feature vector, which was then given to a support vector machine (SVM) classifier.

Bhatia and Indian[9] developed “Tarang”, a feature extraction technique for recognizing and improving offline handwritten Hindi “SWARs” accuracy. Three feature extraction techniques were implemented to determine the feature of each sample image from the dataset of 1950 samples. The recognition rate increases to 95.7% when both local and global wave features are combined.

Puri and Singh[10] developed a novel offline Hindi handwritten document classification system (HHDCS). The Normal-Moderate-Complex (N-M-C) handwriting classification model found that N handwriting performs better than M and C handwriting and uses the right spaces to produce positive recognition results.

Rastogi et al.[11] utilized normalized chain code and gradient direction methods for producing the feature vector of Gujarati numeral images and then trained it through a feed-forward back propagation neural network with the Levenberg-Marquartdt function. A dataset of approximately 2500 samples was used.

Acharya et al.[5] created a new dataset, DHCD, which consists of 92,000 images. There are 46 characters in the Devanagari script, which makes it publicly available for any researcher. According to the experimental findings, CNNs with a dropout layer and a dataset augmentation method can produce extremely high accuracy for testing, even for complex and varied datasets.

Krizhevsky et al.[12] have trained a large deep CNN for classifying 1.2 million high-resolution images into thousands of distinct classes during the ImageNet LSVRC-2010 competition. The neural network (NN) is built with five convolution layers. Some convolution layers were succeeded by max-pool layers, three fully connected layers, and at last 1000-way softmax layer. The dropout regularization technique was used to minimize the overfitting problem significantly.

Dokare et al.[13] explored using a CNN in this study to recognize Devanagari characters. The complexity of applications like character recognition, which require a huge amount of data, can best be handled by deep learning. The recognition accuracies for Devanagari consonants, vowels, and numbers are 98%, 97.56%, and 99%, respectively.

Bisht and Gupta[14] proposed two CNN-based models for recognizing the Devanagari-modified characters. The accuracy of a single CNN architecture under six-fold cross-validation and in tests is 81.52% and 81.62%, respectively. Stage-1 and Stage-2 validation accuracy for the double-CNN architecture were reported at 89.80% and 85.65%, respectively.

Roy et al.[15] have described their work as creating a dynamic programming-based method for recognizing city names and PIN codes in destination addresses on Indian mailing documents. Trilingual city name recognition yielded a 0.20% error rate and a 28.11% rejection rate, whereas handwritten pin code recognition yielded a 0.83% error rate and a 15.27% rejection rate.

Sharma et al.[16] used CNN to recognize city names in the postal automation field. The model was trained and validated at different hyper-parameters on a dataset of 4000 samples from 10 classes in the Gurumukhi script. An Adam optimizer with batch size four and a learning rate of 0.001 gave the best average validation accuracy of 99.13% compared to the stochastic gradient descent (SGD) optimizer.

In addition to English, Roy et al.[17] suggested recognizing the city names, which are handwritten in Bangla and Devanagari script. This study addresses recognizing city names written in trilingual form using deep learning without script identification. A dataset with 24,460 samples collected from 391 cities was used for this. The accuracy rates for Devanagari, Bangla, and English scripts are 93.29%, 96.27%, and 98.01%, respectively.

Qureshi et al.[3] proposed converting the offline handwritten texts written on ruled-line pages into digital text. A custom dataset was created by scanning 400 forms (sentences are from the IAM dataset) with 300 dpi resolution and storing them as png files. Three experiments were performed to evaluate the overall performance of the proposed method. The suggested method attained 26% improved accuracy in the simple HTR case and 20% improved accuracy in the MXNET case.

Aneja and Aneja[18] proposed CNN and transfer learning for handwriting recognition. The "develop model approach" or "pre-trained approach" both use a deep learning method known as "transfer learning". Fine-tuning and ConvNet (fixed feature extractors) are used in transfer learning. The dataset contains 46 different classes, each with 2000 images. Inception, Vgg, AlexNet, and DenseNet are the models ranked from best to worst based on their accuracy levels.

The advantages of BN, which Bjorck et al.[19] have studied, were mostly mediated by higher learning rates, and they contended that the increased implicit regularization of SGD, which enhances generalization, results from the higher learning rate. This research demonstrated that significant parameter adjustments to large learning rates were constrained by the potential for un-normalized networks to produce activations whose magnitudes expand drastically with depth.

The study of Garbin et al.[20] revealed that deep neural network (DNN) training generally uses BN and dropout to enhance the model's performance. Including BN in CNN improves performance without other observable side effects, whereas including dropout in CNN reduces accuracy significantly. BN should be one of the initial steps to optimize a CNN, whereas dropout requires careful consideration as a cautionary sign.

Li et al.[21] used dropout layers in conjunction with batch normalization. They discovered that a neural variance would be incorrect and displaced when information flows in inference due to their different test strategies in CNN architecture. These insights can be used as practical guidance for improving deep learning procedures. The above-proposed systems can be expanded to recognize and classify modified characters and half-characters, image-based words, fonts, italicized text, and imaged documents, as well as numbers with certain digits. They can also be used to recognize scripts with a higher level of complexity, such as compound characters. Deep CNN has been observed as a system for recognizing Devanagari characters, numerals, and modified characters with satisfactory accuracy.

This brief survey concludes that various methods have been employed to solve the OHDCR problem. Deep learning is a technology making its way into the field of text recognition. DNN training is complicated because the distribution of each layer's inputs varies in training as the parameters of the preceding layers change. Hence, it requires lower learning rates, which slows down training. It takes work to train models with saturating nonlinearities known as the internal covariate shift, and it can be solved by normalizing layer inputs. The model's strength should include normalization within the architecture and execution of normalization for each training mini-batch. The dropout requirement can be eliminated by using BN as a regularizer in DNN, which improves accuracy irrespective of the dataset size with a higher learning rate and fewer number epochs. In this study, the use of BN is analyzed in the feature extraction phase only, in the classification phase only, and in both phases of CNN. In **Tables 1** and **2**, present the comparative analysis of various character recognition approaches using different datasets of Devanagari script and non-indic script respectively.

Table 1. Comparison of character recognition approaches using different datasets of Devanagari script.

S. No.	Author	Approach	Dataset	Number of classes	Accuracy	Year
1.	Acharya S et al.[5]	4 layer CNN	DHCD	46	98.47%	2015
2.	Jangid M and Srivastava S[22]	Layer wise deep CNN and different adaptive gradient methods	Isidchar and V2DMDCHAR	47	98%	2018
3.	Deore SP and Pravin A[23]	Fine-tuned VGG 16 architecture	Own newly created	58	96.55%	2020
4.	Mhapsekar M et al.[24]	ResNet 34 and ResNet 50 compared with 4 layer CNN and 8 layer CNN	DHCD	46	ResNet 50 = 99.35%	2020
5.	Gurav Y et al.[25]	Image processing and deep learning	Own character dataset without shirorekha	30	99.65%	2020
6.	Dokare I et al.[13]	4 layer CNN	DHCD (consonants)	36	96.86%	2021
			DHCD (numerals)	10	99.29%	2021
7.	Manocha SK and Tewari P[26]	CNN as feature extractor with different classifiers	DHCD	46	CNN + SVM – RBF = 99%	2021
8.	Mishra M et al.[27]	Bottleneck version of the residual module (ResNet with 85 convolution layer)	DHCD	46	99.72%	2021
9.	Pande SM and Jha BK[28]	Machine learning classifiers like extra trees, random forest, decision tree, KNN, etc.	Own character dataset	43	Extra tree classifier = 78%	2021
10.	Sachdeva J and Mittal S[29]	Edge histogram technique with different machine learning techniques	Own compound character set	50	SVM = 99.88%	2021

Table 2. Comparison of character recognition approaches using different datasets of non-indic script.

S. No.	Author	Script	Approach	Dataset	Accuracy	Year
1.	Sousa Neto AF et al.[30]	English, French and Latin (9th century)	Gated-CNN-BGRU model is motivated by the Bluche model and Puigcerver model	Bentham, IAM, RIMES, saint gall and Washington	Outperformed existing HTR systems by an average of 33% on five handwritten benchmark datasets	2020
2.	Manchala SY et al.[31]	English	NN (5 layers CNN, 2 layers RNN and CTC) and tensorflow	IAM	Above 90.3%	2020
3.	Sree A et al.[32]	English	CNN, RNN, Android app using kivy and kivy MD, SQL alchemy for database storage	IAM	Proposed method 83% and east text detector 46%	2021
4.	Gupta N and Liu W[33]	English	Adaptive line segmentation scheme from unconstrained document image using MATLAB R 2014 b version	Own (Dataset I and Dataset II), ICDAR09, IAM	Own = 98.01%, IAM = 91.99% and ICDAR = 96%	2021
5.	Wang Y et al.[34]	Barcelona, English, Chinese	Offline HTR uses a variety of deep learning techniques for character, word or line and multi-lines recognition	BH2M, IAM, Bentham, HIT-MW, CASIA-HWDB	Current contributions to the offline HTR domain can be categorized into two: HTR with minimal supervision and HTR module that is quicker and smaller	2021

Table 2. (*Continued*).

S. No.	Author	Script	Approach	Dataset	Accuracy	Year
6.	Mondal R et al.[35]	English	YOLOv3 object recognition model trained using darknet framework	IAM	29.21% WER and 9.53% CER	2022
7.	Kumari L et al.[36]	English and German	LexiconNet	IAM, RIMES and READ-2016	Average accuracy increased by 35.10% on IAM, 48.54% on RIMES and 39.79% on READ-2016 from previous methods	2022
8.	AlJarrah MN et al.[37]	Arabic	CNN	AHCD	97.2%	2021
9.	Alkhateeb JH et al.[38]	Arabic	CNN	AHCR, AHCD, and Hijja	89.8%, 95.4%, and 92.5%	2021
10	Nayef BH et al.[39]	Arabic	CNN with optimized leaky ReLU	AHCD, Hijja and self-collected	99%, 90% and 95.4%	2021

3. Proposed methodology

3.1. Dataset

DHCD[5] is a large dataset of the Devanagari character images written by different persons and is widely used by researchers for recognizing handwritten characters. This dataset is openly available at https://archive.ics.uci.edu/ml/datasets/Devanagari+Handwritten+Character+Dataset. The DHCD contains 46 classes, of which 10 are numerals and 36 are consonants. The DHCD does not include vowels. The DHCD has already undergone preprocessing. Each character image is resized to a size of 28 by 28 pixels with a padding of 2 pixels. Padding makes dataset images have a size of 32 by 32 pixels. Images are grayscaled; after this, the intensity of the characters is reversed. Random samples of numerals and consonants taken from the DHCD dataset with assigned class labels are shown in **Tables 3** and **4**.

Table 3. DHCD numerals sample with assigned class labels.

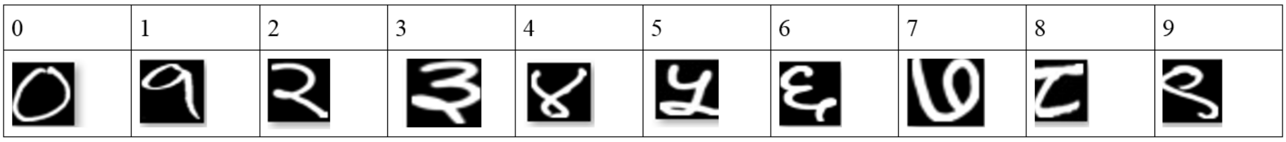
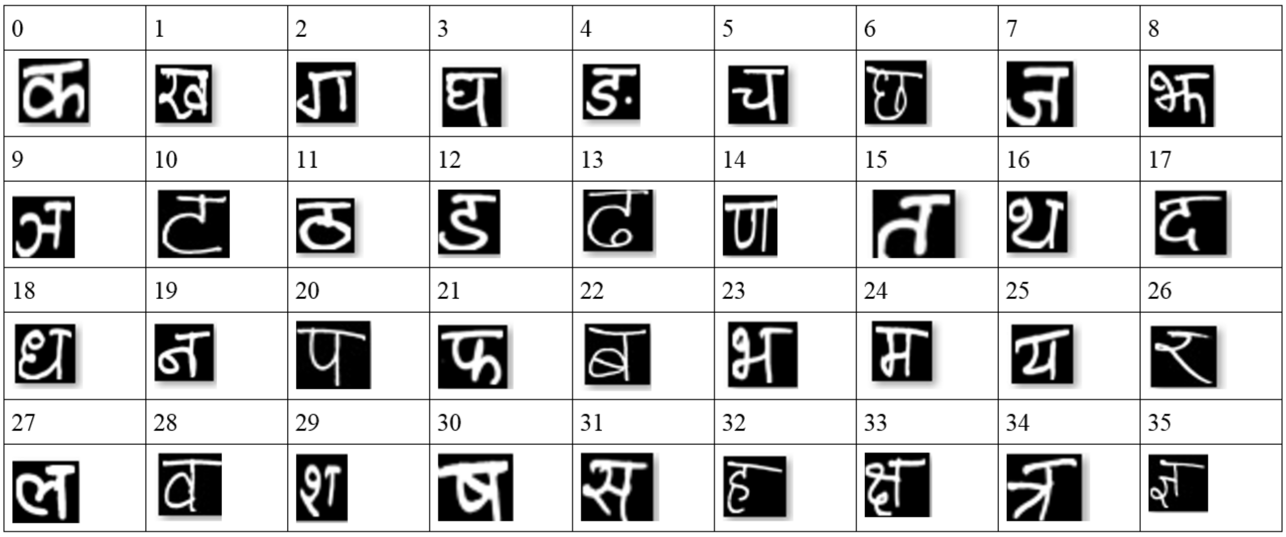


Table 4. DHCD consonants sample with assigned class labels.



3.2. Convolutional neural networks (CNNs)

As mentioned in the studies of Indian and Bhatia[6] and Yamashita et al.[40], CNN is a type of artificial neural network that has been used a lot in computer vision tasks and is the deep learning model with the most well-known technique. CNN is a type of deep learning model used to process data with a grid pattern, like images. CNN is a mathematical model that is usually made up of three types of layers: convolution, pooling, and fully connected layers. The first two layers, convolution and pooling, extract features. The third layer, a fully connected layer, maps the features that were extracted into the end output, which leads to classification.

As CNN-based models effectively extract features, they are utilized to resolve image classification issues. The convolution layer (CL), pooling layer (PL), and fully-connected layer (FCL) are the building blocks of any CNN model. The CNNs overall architecture is designed when these layers are combined. The activation function (AF) and the dropout layer are two more important elements.

The first layer, the CL, is employed to distinguish the different highlights from a given input image. Moving the channel over the input image yields the channel's dot product and the input image's various components in terms of channel approximation. As mentioned in the study of Guha et al.[41], each CL output can be expressed using Equation (1),

$$Out = \frac{(L_{in} + 2 \times Pad - F\ \ )}{S} + 1 \tag{1}$$

where, Out = size of the output, L_{in} = size of the input, Pad = padding size, F = filters size, and S = size of stride to slide the filter.

As mentioned in the study of Guha et al.[41], a CL has three dimensions in,

$$\text{Input} = (H_{in} \times W_{in} \times C_{in}) \tag{2}$$

where, H_{in} = input height, W_{in} = input width and C_{in} = input channels. Each layer in CNN architecture has same calculation for output feature. By using Equation (3), neurons, parameters and connections are produced by CLs,

$$P = W_t + B \tag{3}$$

where, P = parameters, B = bias and W_t = CLs weight calculated using Equation (4),

$$W_t = C_{out} \times (H_{in} \times W_{in}) \times C_{in} \tag{4}$$

where, C_{out} = previous layer's output channel.

The second layer, known as the PL, is employed to map features: max pooling gives the highest value from the part of the image that the kernel covers whereas; average pooling gives the arithmetic mean of all the values from the part of the image that the kernel covers.

As mentioned in the study of Guha et al.[41], the PL with M* M size filters is applied with a stride expressed in Equations (5) and (6),

$$W_{out} = \frac{(W_{in} - F)}{S} + 1 \tag{5}$$

$$H_{out} = \frac{(H_{in} - F)}{S} + 1 \tag{6}$$

where, W_{out} = output width, W_{in} = input width, F = filters, S = stride, H_{out} = output height and H_{in} = input height.

The FCL, the third layer, flattens the features received from the CL and PL.

The BN layer, which normalizes the input of all network layers, is used instead of the dropout layer in addition to CL, PL, and FCL, considerably reducing the training time. Deep neural networks' intermediary layers can have their activations normalized using the BN method. BN has been a preferred deep learning approach due to its propensity to speed up training and increase accuracy.

The CNN model completes with the AF. Any variable-to-variable relationship in a network may be learned and estimated using the AF. The two AFs used in the proposed models are the rectified linear unit (ReLU) and softmax (SM).

As mentioned in the study of Romanuke[42], ReLU employs the non-saturating AF and sets negative values to zero, effectively removing them from an activation map as expressed in Equation (7).

$$f(x) = \max\{0, x\} \tag{7}$$

SM computes probability distributions from a vector of real numbers, as specified in the study of Nwankpa et al.[43]. The resulting output falls within the 0 to 1 value range, with a probability sum of 1. It is used for multi-class models, returning the probabilities of each class, with the highest value going to be the resultant class.

$$f(\mathrm{x_i}) = \frac{\exp(\mathrm{x_i})}{\sum_{\mathrm{j}} \exp(\mathrm{x_j})} \tag{8}$$

For classifying multiple classes, the output layer uses the SM and AF, while the input layer and hidden layers use the ReLU and AF.

3.2.1. Introduction to batch normalization and dropout

Dropout[44] is a method for preventing overfitting. Its core concept is to take an overfitting model and then train sub-models by randomly pruning units from all training batches. Dropout pushes units to be more resilient by continually removing arbitrary units, forcing them to learn features independently without relying on other units. It may be considered a simplified model ensembling in this context. The dropout rate, a new hyper-parameter, governs the number of units to keep in the NN.

BN was developed to address the unpredictability of NN and accelerate learning. A well-known strategy is to normalize the values of each sample before feeding it to the neural network as input. BN takes one step further by normalizing all network layers, not just the input layer. For each mini-batch, the normalization is computed. This normalization enables greater learning rates during training[45].

As in the study of Bjorck et al.[19], BN is generally considered for CNN and computed using Equation (9). The BN layer's output and input are four-dimensional tensors known as $O_{b,c,x,y}$ and $I_{b,c,x,y}$ respectively. The dimensions correspond to examples inside a batch b, channel c, and two spatial dimensions, x and y. BN uses the same normalization for all channel activations.

$$\mathrm{O_{b,c,x,y}} \leftarrow \gamma_{\mathrm{c}} \frac{\mathrm{I_{b,c,x,y}} - \mu_{\mathrm{c}}}{\sqrt{\sigma_{\mathrm{c}}^2 + \epsilon}} + \beta_{\mathrm{c}} \qquad \forall \mathrm{b, c, x, y} \tag{9}$$

in Equation (9), BN subtracts the mean activation,

$$\mu_{\mathrm{c}} = \frac{1}{|\mathrm{B}|} \sum_{\mathrm{b,x,y}} \mathrm{I_{b,c,x,y}} \tag{10}$$

From all input activations in channel c, B contains all channel c activations across all features b in the mini-batch and all spatial x and y locations. In BN, the centered activation is divided by the standard deviation σ_c (plus ϵ for numerical stability), which is derived in the same way. Running mean and variance averages are employed throughout testing. Normalization is then followed by a channel-wise affine transformation that is parameterized via γ_c and β_c, which is learned during training.

3.3. Proposed HDevChaRNet: An overview

The proposed CNN model is briefly explained in this section, which is used to classify Devanagari characters using the DHCD dataset. An outline of the proposed model is presented in **Figure 2**.

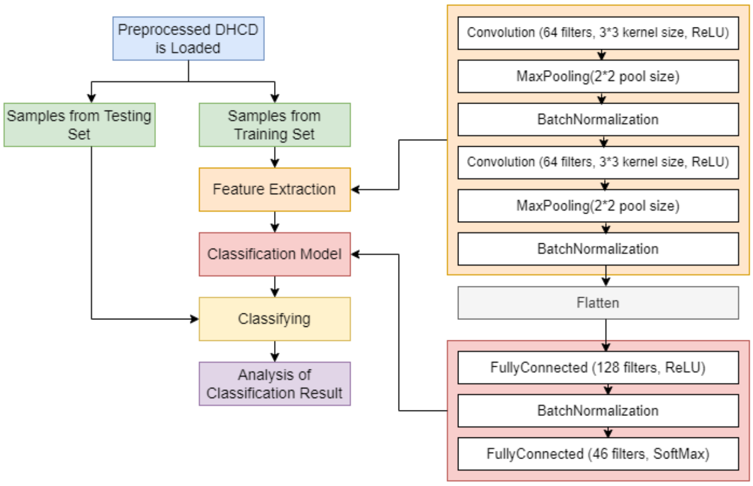


Figure 2. Outline of proposed HDevChaRNet model.

3.3.1. Proposed HDevChaRNet: Architecture

The DHCD consists of 32 × 32 grayscale images of characters and is one of the best-known datasets of Devanagari handwritten numerals and consonants. Three CNN models are proposed in our study to recognize the Devanagari characters. Except for the kernel size, pool size, and activation function, each CNN model has a different number of CLs. Three strategies are proposed to study the application of BN in the CNN model for recognizing the offline handwritten Devanagari characters.

1) FEP: Dropout and BN layers are used at the feature extraction phase of the CNN model.
2) CP: Dropout and BN layers are used at the classification phase of the CNN model, and
3) FECP: Dropout and BN layers are used at both feature extraction and classification phase of the CNN model.

M1, M2, and M3 are the three proposed models with variations in the number of layers, number of filters, and number of neurons, as shown in **Table 5**. All these models employ the Adam optimizer with a default learning rate of 0.001 and a batch size of 200. All these models are employed for three output classes: 46 for both consonants and numerals, 36 for consonants only, and 10 for numerals only.

Table 5. Proposed architecture of model M1, M2 and M3 without dropout and BN Layer.

M1	M2	M3
Input: 32 × 32 × 1 Output: 46/36/10 classes, soft max	Input: 32 × 32 × 1 Output: 46/36/10 classes, soft max	Input: 32 × 32 × 1 Output: 46/36/10 classes, soft max
Conv2D (32, kernel size = 3, ReLU) Conv2D (64, kernel size = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Conv2D (128, kernel size = 3, ReLU) Conv2D (256, kernel size = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Conv2D (512, kernel size = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Flatten () Dense (128, ReLU) Dense (64, ReLU)	Conv2D (64, kernel size = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Conv2D (64, kernel size = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Conv2D (64, kernelsize = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Flatten () Dense (64, ReLU)	Conv2D (64, kernel size = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Conv2D (64, kernel size = 3, ReLU) MaxPool2D (pool size = 2, strides = 2) Flatten () Dense (128, ReLU)

4. Results analysis

The results of all proposed models are presented in **Tables 6–8** and discussed as follows:

1) The M1 model without dropout and BN layer has the highest testing accuracy of 98.14%, 98.08%, and 99.50% for 46, 36, and 10 output classes, respectively, as shown in **Table 6**.

Table 6. Performance of proposed HDevChaRNet models without dropout and BN layer.

Batch size = 200	**M1**		**M2**		**M3**	
Number of output classes	**Training accuracy**	**Testing accuracy**	**Training accuracy**	**Testing accuracy**	**Training accuracy**	**Testing accuracy**
46	99.76	98.14	99.69	97.93	99.79	97.71
36	99.55	98.08	99.42	97.39	99.93	97.07
10	99.98	99.50	100	99.43	100	99.33

2) The M1 model with BN layer has the highest testing accuracy of 99.17%, 98.75%, and 99.70% for 46, 36, and 10 output classes, respectively, as shown in **Table 7**.

3) When the results of **Table 6** and **Table 7** are compared it is found that M1 model has highest testing accuracy in both tables. The only difference is that M1 model without BN and dropout has more overfitting as compared to M1 model with BN.

4) Each of the three models has the highest training accuracy of 100% for 10 output classes in all three phases of each model, as shown in **Table 7**.

5) For 46 output classes, the CP of models M1 and M2 has the highest training and testing accuracy, whereas the FECP of model M3 has the highest training and testing accuracy, as shown in **Table 7**.

6) The FECP of each of the three models has the highest training and testing accuracy for the 36 output classes among all three phases of each model, as shown in **Table 7**.

7) The CP of each of the three models has the highest training and testing accuracy for 10 output classes among all three phases of each model, as shown in **Table 7**.

Table 7. Performance of proposed HDevChaRNet models with BN layer.

Accuracy																		
	M1						**M2**						**M3**					
	FEP		**CP**		**FECP**		**FEP**		**CP**		**FECP**		**FEP**		**CP**		**FECP**	
Number of output classes	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**
46	99.81	98.70	100	99.17	99.92	98.91	99.91	98.46	100	98.80	100	98.79	99.80	98.15	100	98.80	100	98.93
36	99.50	98.40	99.64	98.17	99.89	98.75	99.99	98.43	98.13	96.24	100	98.72	99.92	98.08	100	98.58	100	98.73
10	100	99.60	100	99.70	100	99.57	100	99.57	100	99.69	100	99.63	100	99.47	100	99.60	100	98.57

8) FECP of model M1 with dropout layer has the highest testing accuracy of 98.86%, 98.80%, and 99.57% for 46, 36, and 10 output classes, whereas FEP of model M3 has the highest training accuracy of 99.81%, 99.70%, and 99.90% for 46, 36, and 10 output classes, as shown in **Table 8**.

9) In model M1, FECP has the highest training and testing accuracy for all output classes as compared to FEP and CP, as shown in **Table 8**.

10) In models M2 and M3, FEP has the highest training accuracy for all output classes as compared to CP and FECP, whereas FECP has the highest testing accuracy for all output classes as compared to FEP and CP, as shown in **Table 8**.

Table 8. Performance of proposed HDevChaRNet models with dropout layer.

Accuracy																		
	M1						**M2**						**M3**					
	FEP		**CP**		**FECP**		**FEP**		**CP**		**FECP**		**FEP**		**CP**		**FECP**	
Number of output classes	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**	**Training**	**Testing**
46	99.31	98.74	99.20	98.36	99.41	98.86	99.05	98.64	98.99	98.33	98.06	98.71	99.81	98.00	99.20	98.38	99.37	98.75
36	98.97	98.31	98.67	97.96	99.02	98.80	99.26	98.48	99.24	98.04	98.42	98.69	99.70	97.80	98.66	97.79	99.40	98.46
10	99.66	99.37	99.61	99.43	99.67	99.57	99.84	99.47	99.59	99.47	99.29	99.50	99.90	99.30	99.85	99.47	99.73	99.53

1) The accuracy graph shown in **Figure 3** depicts that model M1 without dropout and BN has the highest testing accuracy for 10, 36 and 46 output classes.

2) The accuracy graph shown in **Figure 4** depicts that model M1 with BN at CP has the highest testing accuracy for 46 and 10 output classes.

3) The accuracy graph shown in **Figure 5** depicts that model M3 with BN at FECP has the second-highest testing accuracy for 46 and 36 output classes, respectively.

4) The accuracy graph shown in **Figure 6** depicts that model M2 with BN at CP has the second highest testing accuracy for 10 output classes and has more overfitting for 46 and 36 output classes.

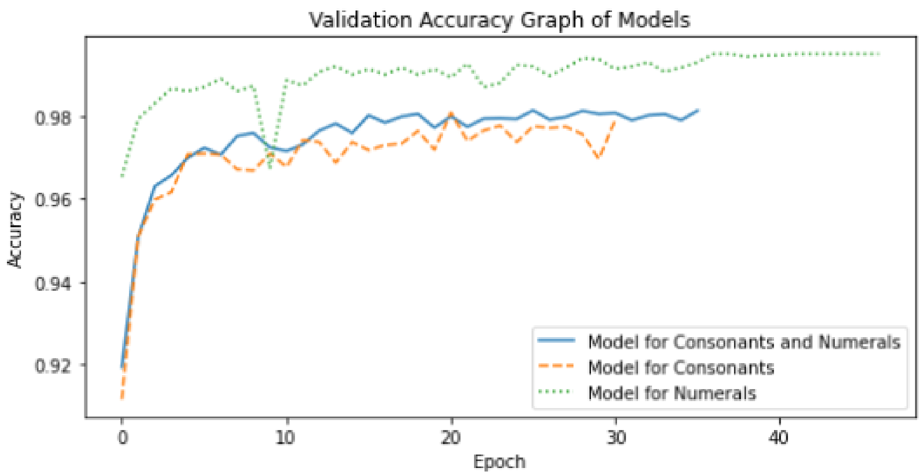


Figure 3. Accuracy graph of proposed model HDevChaRNet M1 without dropout and BN.

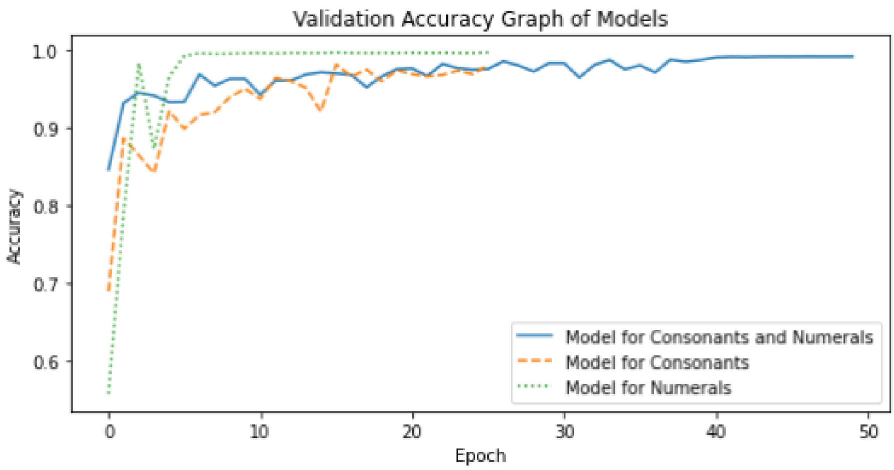


Figure 4. Accuracy graph of proposed model HDevChaRNet M1 with BN layer at CP.

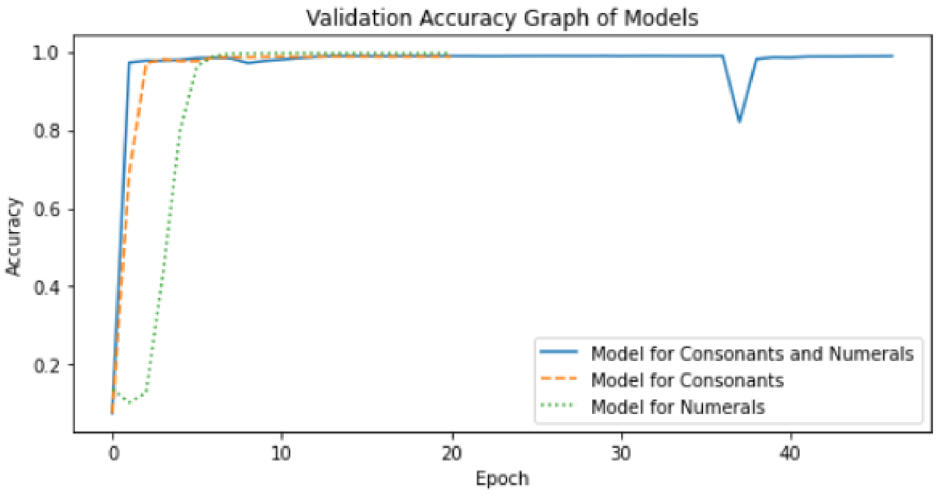


Figure 5. Accuracy graph of proposed model HDevChaRNet M3 with BN layer at FECP.

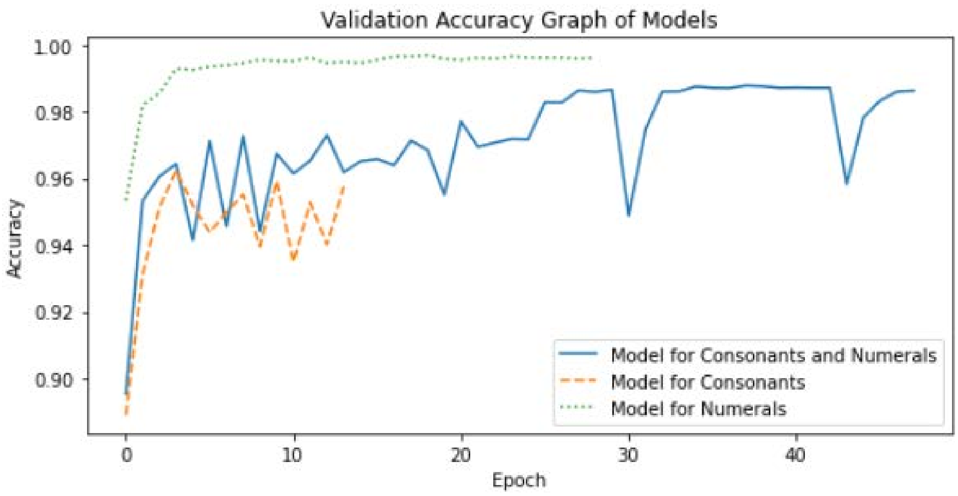


Figure 6. Accuracy graph of proposed model HDevChaRNet M2 with BN layer at CP.

5) On comparing the performance of all models as shown in **Tables 6–8**, it is clear that the models with BN layers are giving better results as compared to models with dropout, without dropout and BN.

6) In **Table 9**, comparisons of performances of the proposed models with other state-of-the-art are presented and observed the following outcomes:

• For 46 distinct characters (both consonants and numerals), the proposed HDevChaRNet model M1 with batch normalization at classification phase has attained an accuracy of 99.17%, which is higher compared to the 98%, 98.47% and 99% attained by Acharya et al.[5], Aneja and Aneja[18] and Manocha and Tewari[26], respectively.

• The proposed HDevChaRNet model M1 with batch normalization at feature extraction phase and classification phase has also attained a better accuracy of 98.75% compared to the 96.86% attained by Dokare et al.[13] for 36 distinct characters (consonants only).

• The proposed HDevChaRNet model M1 with batch normalization at classification phase has attained a better accuracy of 99.70% compared to the 99.29% attained by Dokare et al.[13] for 10 distinct characters (numerals only).

Table 9. Comparisons of performances of the proposed HDevChaRNet models with other states-of-the-art.

Authors	Approach	Dataset used	Character set used from dataset	No. of samples used from dataset	No. of distinct characters labels	Batch size	Accuracy (%)
Aneja N and Aneja S[18]	AlexNet, Vgg16 and Vgg19	DHCD	Consonants and numerals	92,000	46	32	98
Acharya S et al.[5]	4 layer CNN	DHCD	Consonants and numerals	92,000	46	200	98.47
Manocha SK and Tewari P[26]	CNN as feature extractor with different classifiers	DHCD	Consonants and numerals	92,000	46	-	99
Proposed HDevChaRNet CP of M1 using BN	8 layer CNN	DHCD	Consonants and numerals	92,000	46	200	99.17
Dokare I et al.[13]	4 layer CNN	DHCD	Consonants	72,000	36	200	96.86
	4 layer CNN	DHCD	Numerals	20,000	10	200	99.29
Proposed HDevChaRNet FECP of M1 using BN	8 layer CNN	DHCD	Consonants	72,000	36	200	98.75
Proposed HDevChaRNet CP of M1 using BN	8 layer CNN	DHCD	Numerals	20,000	10	200	99.70

5. Future direction

Due to the non-availability of standard public datasets for offline handwritten Devanagari characters, DHCD is being used for the proposed models. This dataset comprises consonants and numerals only, not vowels. So in the future, these proposed models can also be applied to datasets consisting of vowels, and the results can be compared with other state-of-the-art ones. As the proposed models are limited to the recognition of individual characters without modifiers, the work can be further extended to recognize the characters with modifiers or words or text of Devanagari.

6. Conclusion

Various experiments have been carried out at different phases (FEP, CP, and FECP) of the CNN architecture to demonstrate the main benefit of BN. It is found that BN allows training at a higher learning rate, leading to faster convergence and greater generalization as compared to dropout for recognizing Devanagari handwritten characters. Three models are proposed, and each of these has a different number of layers except the kernel size and pool size. Results are analyzed in three parts: neutral cases of models (where neither BN nor dropout layer is used); models using BN at different phases; and models using dropout at different phases. Models using BN have attained the highest accuracy among all the results when analyzed.

Data availability

The dataset used in this research is openly available at https://archive.ics.uci.edu/ml/datasets/Devanagari+Handwritten+Character+Dataset.

Author contributions

Conceptualization, BY and AI; methodology, BY; software, BY; validation, BY, AI and GM; formal analysis, BY; investigation, BY; resources, BY and AI; data curation, BY; writing—original draft preparation, BY; writing—review and editing, BY, AI and GM; visualization, BY; supervision, AI.

Conflict of interest

The authors declare no conflict of interest.

References

1. Bal A, Saha R. An improved method for handwritten document analysis using segmentation, baseline recognition and writing pressure detection. *Procedia Computer Science* 2016; 93: 403–415. doi: 10.1016/j.procs.2016.07.227
2. Bozinovic RM, Srihari SN. Offline cursive script word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1989; 11(1): 68–83. doi: 10.1109/34.23114
3. Qureshi F, Rajput A, Mujtaba G, Fatima N. A novel offline handwritten text recognition technique to convert ruled-line text into digital text through deep neural networks. *Multimedia Tools and Applications* 2022; 81(13): 18223–18249. doi: 10.1007/s11042-022-12097-7
4. Puri S, Singh SP. An efficient devanagari character classification in printed and handwritten documents using SVM. *Procedia Computer Science* 2019; 152: 111–121. doi: 10.1016/j.procs.2019.05.033
5. Acharya S, Pant AK, Gyawali PK. Deep learning based large scale handwritten devanagari character recognition. In: Proceedings of 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA); 15–17 December 2015; Kathmandu, Nepal.
6. Indian A, Bhatia K. A survey of offline handwritten hindi character recognition. In: Proceedings of 2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA); 15–16 September 2017; Dehradun, India. pp. 1–6.
7. Bhalerao M, Bonde S, Nandedkar A, Pilawan S. Combined classifier approach for offline handwritten Devanagari character recognition using multiple Features. In: *Computational Vision and Bio Inspired Computing*, 1st ed. Springer; 2018. pp. 45–54.
8. Ghosh R, Panda C, Kumar P. Handwritten text recognition in bank cheques. In: Proceedings of 2018 Conference on Information and Communication Technology (CICT'18); 26–28 October 2018; Jabalpur, India.
9. Bhatia K, Indian A. Offline handwritten hindi 'SWARs' recognition using a novel wave based feature extraction method. *International Journal of Computer Science Issues* 2017; 14(4): 8–14. doi: 10.20943/01201704.814
10. Puri S, Singh SP. Toward recognition and classification of hindi handwritten document image. In: *Ambient Communications and Computer Systems*, 1st ed. Springer; 2019. pp. 497–507.
11. Rastogi N, Dutta M, Indian A. Offline handwritten numerals recognition using combinational feature extraction approach. *International Journal of Innovative Research in Computer and Communication Engineering* 2017; 5(4): 7844–7851.
12. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* 2012; 25(2): 84–90. doi: 10.1145/3065386
13. Dokare I, Gadge S, Kharde K, et al. Recognition of handwritten devanagari character using convolutional neural network. In: Proceedings of 2021 3rd International Conference on Signal Processing and Communication (ICPSC); 13–14 May 2021; Tamil Nadu, India.
14. Bisht M, Gupta R. Offline handwritten devanagari modified character recognition using convolutional neural network. *Sadhana* 2021; 46(1). doi:10.1007/s12046-020-01532-w
15. Roy RK, Pal U, Roy K, Kimura F. A system for recognition of destination address in postal documents of India. *Malaysian Journal of Computer Science* 2020; 3(3): 202–216. doi: 10.22452/mjcs
16. Sharma S, Gupta S, Gupta D, et al. Recognition of gurumukhi handwritten city names using deep learning and cloud computing. *Scientific Programming* 2022; 2022. doi: 10.1155/2022/5945117
17. Roy RK, Mukherjee H, Roy K, Pal U. CNN based recognition of handwritten multilingual city names. *Multimedia Tools and Applications* 2022; 81(8): 11501–11517. doi: 10.1007/s11042-022-12193-8
18. Aneja N, Aneja S. Transfer learning using CNN for handwritten devanagari character recognition. In: Proceedings of 2019 1st International Conference on Advances in Information Technology (ICAIT); 25–27 July 2019; Chikmagalur, India.

19. Bjorck J, Gomes C, Selman B, Weinberger KQ. Understanding batch normalization. In: Proceedings of 32nd Conference on Neural Information Processing Systems (NeurIPS 2018); 2–8 December 2018; Montréal, Canada.
20. Garbin C, Zhu X, Marques O. Dropout vs. batch normalization: An empirical study of their impact to deep learning. *Multimedia Tools and Applications* 2020; 79(19–20): 12777–12815. doi: 10.1007/s11042-019-08453-9
21. Li X, Chen S, Hu X, Yang J. Understanding the disharmony between dropout and batch normalization by variance shift. In: Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 15–20 June 2019; Long Beach CA, USA.
22. Jangid M, Srivastava S. Handwritten devanagari character recognition using layer-wise training of deep convolutional neural networks and adaptive gradient methods. *Journal Imaging* 2018; 4(2): 41. doi: 10.3390/jimaging4020041
23. Deore SP, Pravin A. Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset. *Sadhana* 2020; 45(1): 243. doi: 10.1007/s12046-020-01484-1
24. Mhapsekar M, Mhapsekar P, Mhatre A, Sawant V. Implementation of residual network (ResNet) for devanagari handwritten character recognition. In: Vasudevan H, Michalas A, Shekokar N, et al. (editors). *Advanced Computing Technologies and Applications*. Springer, Singapore; 2020.
25. Gurav Y, Bhagat P, Jadhav R, Sinha S. Devanagari handwritten character recognition using convolutional neural networks. In: Proceedings of 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE); 12–13 June 2020; Istanbul, Turkey. pp. 1–6.
26. Manocha SK, Tewari P. Devanagari handwritten character recognition using CNN as feature extractor. In: Proceedings of 2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON); 29–30 October 2021; Pune, India. pp. 1–5.
27. Mishra M, Choudhury T, Sarkar T. Devanagari handwritten character recognition. *2021 IEEE India Council International Subsections Conference (INDISCON)* 2021; 1–6. doi: 10.1109/INDISCON53343.2021.9582192
28. Pande SM, Jha BK. Character recognition system for devanagari script using machine learning approach. In: Proceedings of 2021 5th International Conference on Computing Methodologies and Communication (ICCMC); 8–10 April 2021; Erode, India. pp. 899–903.
29. Sachdeva J, Mittal S. Handwritten offline devanagari compound character recognition using machine learning. In: Proceedings of ACI'21: Workshop on Advances in Computational Intelligence at ISIC 2021; 25–27 February 2021; Delhi, India.
30. Sousa Neto AF, Bezerra BLD, Toselli AH, Lima EB. HTR-Flor: A deep learning system for offline handwritten text recognition. In: Proceedings of 2020 33rd Sibgrapi Conference on Graphics, Patterns and Images (SIBGRAPI); 7–10 November 2020; Porto de Galinhas, Brazil.
31. Manchala SY, Kinthali J, Kotha K, et al. Handwritten text recognition using deep learning with tensorflow. *International Journal of Engineering Research & Technology (IJERT)* 2020; 9(5). doi: 10.17577/IJERTV9IS050534
32. Sree A, Chennamsetti V, Maliakal DT, et al. Handwriting recognition using CNN and RNN. *Journal of Chengdu University of Technology (Science and Technology Edition)* 2021; 26(7).
33. Gupta N, Liu W. Line segmentation from unconstrained handwritten text images using adaptive approach. *Computer Science* 2021. doi: 10.48550/arXiv.2104.08777
34. Wang Y, Xiao W, Li S. Offline handwritten text recognition using deep learning: A review. *Journal of Physics: Conference Series* 2021; 1848(1): 012015. doi: 10.1088/1742-6596/1848/1/012015
35. Mondal R, Malakar S, Smith EHB, Sarkar R. Handwritten English word recognition using a deep learning based object detection architecture. *Multimedia Tools and Applications* 2022; 81(1): 975–1000. doi: 10.1007/s11042-021-11425-7
36. Kumari L, Singh S, Rathore VVS, Sharma A. A comprehensive handwritten paragraph text recognition system: LexiconNet. *arXiv* 2022; arXiv:2205.11018. doi: 10.48550/arXiv.2205.11018
37. AlJarrah MN, Zyout MM, Duwairi R. Arabic handwritten characters recognition using convolutional neural network. In: Proceedings of 2021 12th International Conference on Information and Communication Systems (ICICS); 24–26 May 2021; Valencia, Spain, pp. 182–188.
38. Alkhateeb JH, Turani A, Abuhamdah A, et al. An effective deep learning approach for improving off-line arabic handwritten character recognition. *International Journal of Software Engineering and Computer Systems* 2021; 6(2): 104–112. doi: 10.15282/ijsecs.6.2.2020.7.0076
39. Nayef BH, Abdullah SNHS, Sulaiman R, Alyasseri ZAA. Optimized leaky ReLU for handwritten arabic character recognition using convolution neural networks. *Multimedia Tools and Applications* 2022; 81: 2065–2094. doi: 10.1007/s11042-021-11593-6
40. Yamashita R, Nishio M, Do RK, Togashi K. Convolutional neural networks: An overview and application in radiology. *Insights into Imaging* 2018; 9(4): 611–629. doi: 10.1007/s13244-018-0639-9
41. Guha R, Das N, Kundu M, et al. DevNet: An efficient CNN architecture for handwritten devanagari character recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 2019; 34(12): 20052009. doi: 10.1142/S0218001420520096

42. Romanuke V. Appropriate number and allocation of RELU in convolutional neural network. *Research Bulletin of the National Technical University of Ukraine Kyiv Politechnic Institute* 2017; 1: 69–78. doi: 10.20535/1810-0546.2017.1.88156
43. Nwankpa C, Ijomah W, Gachagan A, Marshall S. Activation functions: Comparison of trends in practice and research for deep learning. In: Proceedings of 2nd International Conference on Computational Sciences and Technology; 17–19 December 2020; Jamshoro, Pakistan.
44. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 2014; 15(1): 1929–1958. doi: 10.5555/2627435.2670313
45. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning; 6–11 July 2015; Lille, France. pp. 448–456.