



Handwritten Devanagari Character Recognition Using Modified Lenet and Alexnet Convolution Neural Networks

Duddela Sai Prashanth^{1,2} · R. Vasanth Kumar Mehta¹ · Kadiyala Ramana³ · Vidhyacharan Bhaskar⁴ 

Accepted: 8 August 2021 / Published online: 24 September 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

Despite many advances, Handwritten Devanagari Character Recognition (HDCR) remains unsolved due to the presence of complex characters. For HDCR, the traditional feature extraction and classification techniques are limited to the datasets developed in the respective laboratory that are not available publicly. A standard benchmarking dataset is not available for HDCR that helps to develop deep learning models. To progress the performance of HDCR, in this study, we produced a dataset of 38,750 images of Devanagari numerals, and vowels are generated and made publicly available for fellow researchers in this domain. This data is collected from more than 3000 subjects of different age groups. Each character is extracted by a segmentation technique proposed here, which is limited to this application. Experiments are conducted on the dataset; three different Convolution Neural Networks (CNN) architecture is developed. 1. CNN, 2. Modified Lenet CNN (MLCNN) and 3. Alexnet CNN (ACNN). A Modified LCNN is proposed by changing the architecture of Lenet 5 CNN. Regular Lenet 5 has $\tanh(x)$ as its activation function. Since the Devanagari characters are nonlinear, non-linearity is introduced in the Networks by using Rectified Linear Unit. This solves the problem of vanishing gradient problem by $\tanh(x)$. We achieved a recognition rate of 96% on training data and 94% on unseen data using CNN. MLCNN obtained an accuracy rate of 99% and 94% with less computational cost. Whereas, ACNN attained a recognition rate of 99% and 98% on unseen data. A series of experiments were conducted on the data with different combination splits of data and found a minimum loss of 0.001%. Such developments fill a significant percentage of the huge gap between real-world requirements and the actual performance of Devanagari recognizers.

Keywords Devanagari character recognition · Convolution neural network · Computer vision · Pattern recognition

✉ Vidhyacharan Bhaskar
vcharan@gmail.com

Extended author information available on the last page of the article

1 Introduction

In the digital world, the demand for accessing multimedia is emerging. It involves a process of adapting the offline entities into the digital domain. The analysis of the scanned documents deals with the images. In real-time, character recognition on those images is a complex process. The automation of character segmentation and recognition of characters in real-time is a complex process—it includes character segmentation and recognition of characters with the features extracted [1]. Optical Character Recognition (OCR) is categorized into two methods, i.e., online and offline.

Character recognition is easier for the documents with Latin script with the form of the English language. A lot of research is carried for handwritten character recognition in various scenarios like English, Chinese, and Arabic scripts [2]. In any case, for Indian Languages, OCR framework work is as yet slacking when compared with others due to its complicated structure and computations. Devanagari script is base for many languages like Hindi, Marathi, Sanskrit, and many more [3]. It contains 10 numerals, 13 vowels, and 33 consonants. In this research, majorly concentrate on the recognition of offline handwritten Devanagari numeral and vowels.

Offline handwritten Devanagari character recognition (HDCR) is an emerging research field over decades because of the challenges of variant character classes. Despite many advances, HDCR remains unsolved due to the presence of complex characters. Most of the traditional recognizers have not to lead better performance, or it is limited to their dataset, because of not having big datasets as a benchmark. The variety of handwriting styles makes it more difficult due to the similarities between the characters [4]. The progress in this research is improved recognition accuracies with the help of traditional approaches. Though, the number of studies increased—crafted features and robust classifiers still recognition accuracy is far behind the human ability. The reason is the lack of benchmark datasets [5]. Many researchers have developed their dataset in their laboratory and classifiers developed, accuracies are limited to the respective dataset, and most of the datasets are not available publicly [6]. Implementing HDCR systems is much difficult than printed characters, especially for the Indian scripts [7]. The algorithms used for the printed Devanagari characters can often be used on the handwritten characters. Most of the time, it fails to work on the handwritten characters because of the variation in writing style and sizes of different writers.

Most of the OCR techniques developed on HDCR includes a time-consuming pre-processing step. After noise removal and character segmentation, a process that extracts the header line and separates the upper part and lower part of the line for every character. Classifiers are built on the basis of features extracted from these two parts of the character.

The primary objective of this study is to develop a CNN-based model with a lower computational complexity and memory space while maintaining the required accuracy. As a result, the paper is divided into two parts: (a) We investigate and compare/analyze several different state-of-the-art CNN models; and (b) We develop a new CNN model, which we refer to as the "Modified LeNet Convolutional Neural Network (MLCNN)," that requires less time and memory space than existing CNN models. On some publicly available datasets, "MLCNN" is compared to other popular CNN models for validation purposes.

In this paper, a new dataset is presented for Devanagari numerals and vowels. A total of 38,750 images are collected from 2400 subjects of different age groups. This dataset is publicly available [8]. A segmentation algorithm is proposed for the dataset collected, and Convolution Neural Network (CNN) architectures are proposed for the HDCR. First, CNN

obtained 96% accuracy and 94% for unseen data. Secondly, LeNet CNN architecture is modified by introducing max-pooling instead of average pooling to reduce the parameters of fully connected layers then added non-linearity by using Rectified Linear Unit. Modified LeNet Convolution Neural Network (MLCNN) obtained an accuracy of 99% accuracy on Training data and 94% on unseen data. Thirdly, AlexNet CNN (ACNN) architecture achieved an accuracy of 99% and 98% on the unseen dataset and made a detailed comparison between these architectures by separating the data into different ratios of training and test (i.e., 80:20, 70:30, 60:40) with varying epochs of count (i.e., 20, 30, 40 and 50 epochs).

The rest of this paper is organized as follows. Section 2 reviews the related works about Offline HDCR. Section 3 provides the method: the basic theory of CNN, ACNN, and MLCNN and details about the proposed architecture. Section 4 presents the experiment results and discussion that includes the experimental data, settings for the experiment, and training strategy along with the comparison of the results. Finally, the conclusion is given in Sect. 5.

2 Related Work

Offline Handwritten Devanagari Character Recognition (HDCR) has received intense attention from the past few decades. The traditional OCR method includes three stages: pre-processing, feature extraction, and classification. Pre-processing consists of the steps of size normalization and resampling [6]. The variations in the input may adversely impact recognition. Features characterize the best representation of the shape of a character.

Character segmentation is classified into word, line, and character segmentation [1, 42]. It is more accessible to segment printed characters when compared to handwritten characters. There are many standard segmentation techniques developed for printed text. For HDCR, segmentation-based classifiers are popular before pattern-based neural implementation. Different approaches of segmentation as per the dataset is implemented.

Feature extraction techniques developed are of shape-based or non-shape based [10, 41]. These techniques generate a feature vector. These vectors are used as input for any classifiers, unlike deep learning methods, feed a pre-processed image as input along with its label for supervised learning. Hybrid features—a combination of shape-based and non-shape-based features classify the input image more accurately. Still, the automatic features generated by CNN's are better.

In the recent review article [11], structured based and statistical features and their accuracies are compared. Chain code and gradient features achieved an accuracy of 98.51% by the SVM classifier for the printed Devanagari characters [12]. Few Structural features with different classifiers like syntactic pattern analysis-based classifier [13] (90% accuracy), Binary tree classifier [14] (95% accuracy), Distance-based classifier [15] (93% accuracy) for printed text. Different feature extraction techniques implemented for HDCR, quadratic based classifier on the histogram of directional chain code features [16] achieved 98.86% accuracy with the dataset size of 11,270 handwritten Devanagari Characters. The most commonly used classifiers are MLP based classifiers [17–19], SVM [18, 20, 21] Neural Networks [22, 23] or Modified Neural Networks for feature-based classification.

There is no standard dataset that is implemented nor used for Devanagari characters, unlike MNIST for Latin numbers. The dataset size used to develop feature-based classifiers vary. Few datasets developed are 11,270 [16] and achieved an accuracy of 98.86%, 25,000 [18] characters with an accuracy of 90.116% by SVM and 87.56% by MLP classifier.

Zernike Moment based feature [21] on 27,000 handwritten characters with an accuracy of 98.37% by SVM.

Deep learning models has increased its acceptance in the field of Computer Vision and Pattern Recognition due to CNN [24]. CNN includes deep layers of convolutions, subsampling layers, feature extraction over pooling layers, stride, and padding operations to control the size of image representation. It also comprises different activation functions like sigmoid, softmax, to introduce non-linearity—Rectified Linear units, and more. To avoid overfitting, many regularization techniques are included in CNN's. The classification methods are categorized into Kernel based, statistical based and Neural network based [25]. In [37], different language scripts numerals data is used and classified using CNN and able to achieve 99% accuracy. The numerals dataset used in [37] is Arabic (MADBase), MNIST, and Persian. In [38], a comparison of CNN-based classification and feature-based classification is presented. The maximum accuracy able to reach by HoG based features and MLP classifier is 82.66% where as using CNN, 94.91% accuracy, and 96.09% accuracy by Bidirectional Long Short Term Memory (BLSTM). In [39, 40], CNN with linear function— $\tanh(x)$, accuracy achieved is 93.8% and 85.1%. When nonlinearity is introduced to CNN 96.9% accuracy is achieved.

This brief inre survey reveals that there is a lack of standard datasets for Devanagari Scripts. With larger datasets, CNN models can be designed with high accuracies than the classification techniques. Hence, in the present work, we present a dataset and developed a unique segmentation algorithm that is limited to this collected data. Three types of CNN models are developed and compared, CNN, ACNN, and MLCNN.

3 Proposed System

3.1 Preprocessing

The traditional method of recognizing individual characters consists of three steps, pre-processing, feature extraction, and classification. In deep learning models, features are extracted by the Neural model automatically [43]. There is no exclusive process involved, but a few techniques are required in pre-processing. The below-mentioned steps are implemented on the data collected by different subjects, as shown in Figs. 1, and 2. that extracts the images into a folder. Those images are manually identified and separated into respective folders for classification.



Fig. 1 Sample sheet used to collect Devanagari numeral data

The pre-processing – Segmenting character from images

Step 1: Read the scanned images from the folder

Step 2: For each image read, convert the RGB image to grayscale

$$\text{Grayscaleimage} = 0.299R + 0.587G + 0.144B$$

Step 3: Binarize the grayscale image by specifying some threshold in (1)

$$\sigma_B^2(k) = \frac{[\mu_{TW}(k) - \mu(k)]}{w(k)[1 - w(k)]} \quad (1)$$

Step 4: Edge detection is employed to identify the boundaries of the letter

Step 5: Add the pixels to the boundaries of the image using Image dilation in equation (2)

The dilation of A by B is defined by

$$A \oplus B = \bigcup_{b \in B} A_b \quad (2)$$

Step 6: Remove all images connected components having fewer than 10 pixels

Step 7: Label the connected components of the image and store the corresponding binary values in a matrix

Step 8: Measure the properties of each labeled region in the label matrix

Step 9: For each image, do

- I. Plot a bounding box around the recognized letters
 - II. Place the coordinates of the bounding box on the original image and crop the letters
 - III. Resize the cropped images to a dimension of 28×28
-

The size of each original character varies from one image to another. Since the bounding box is plotted on the recognized character and crops it by the original coordinates. Each subject writes in their own ways leads to the variation of the size of characters on the paper. All the characters extracted are resized to 28×28 . Following the standards of one of the most standard and benchmark datasets is MNIST for numerals, the size of the character is chosen as 28×28 .

3.2 Deep Learning Models

From the past few decades, research in Deep Learning models (DLM) has delivered remarkable outcomes in the field of computer vision, machine learning, and pattern recognition. DLM's are more efficient because of their deep architecture. Unlike any conventional classifiers, DLM's accept raw or pre-processed images as inputs. These models do not entail any particular feature extraction. The deep layers, extract features from the input at the lower and middle levels and high-level layers perform classification. These models integrate everything into a single network that can be optimized as per the objective function. These type of integrated models results better than traditional classifiers. The following concepts are being utilized in this research to build such models.

3.3 Convolutions

Convolutions were employed for automatic feature extraction [26]. For an image I in size of (p, q) , the convolution is defined as

$$C(p, q) = (I * K)(p, q) = \sum_k \sum_l I(p - m, q - n)K(m, n) \quad (3)$$

where K is the kernel is the size of (m, n) . Convolution is a mathematical function to describe the process of combining two functions to produce a third function. The output is called a feature map, and by using a filter or kernel to the input image, extract features for training.

3.4 Dropouts and L2 Regularization

Devanagari vowels and numerals are highly sensitive and complex. Characters in Table 5 out of 13 different vowels six vowels are identical with minor changes. Similarly, numerals in Table 6 out of 10 numerals, two couple of numerals, and three numerals are identical with changes. This Features extracted vary with small differences. It leads to over tune of weights to our dataset used for training that may lead to the problem of overfitting in the data, and classifier performs poorly on the unseen data. Regularization methods reduce overfitting [27]. Different regularization methods are L1 & L2 regularization, cross-validation, early stopping, drop out, dataset augmentation. In this research, dropout is adopted and well performed on the data.

Dropout refers to dropping nodes to reduce overfitting. This regularization ignores some neurons in the network during forwarding and backward propagation. This approach helps in reducing interdependent learning among the neurons. CNN extracts the most robust features.

In dropout a parameter ' p ' that sets the probability of which nodes are kept or $(1 - p)$ for those that are dropped.

$$E = \frac{1}{2} \left(t - \sum_{i=1}^n p_i w_i I_i \right)^2 + \sum_{i=1}^n p_i (1 - p_i) w_i^2 I_i^2 \quad (4)$$

L2 regularization is also called Ridge regression. It helps to penalize weights. Any larger weights are gradients with abrupt changes in the model are manifest with a decision boundary by penalizing them it becomes smaller. λ controls the degree of penalty.

$$E = \frac{1}{2} (\text{target}O_1 - \text{out}O_1)^2 + \frac{\lambda}{2} w_i^2 \quad (5)$$

3.5 Subsampling

Subsampling reduces the parameters or the dimensions of the feature map. It helps to discard the spatial information. For input to the further layers in CNN, the parameters are reduced by taking a collection of neurons. In this research, max pooling is used by taking the whole image and is partitioned into rectangles. From each box, a maximum value is taken and forms a new box [28]. Only one value is considered from a group of values. It helps to avoid further overfitting of data.

$$X_{(p,i,j)}^n = f \left(\max_{0 \leq u, v \leq M_n - 1} X_{(p, iS_n + u, jS_n + v)}^{n-1} \right) \quad (6)$$

3.6 Activation Functions

The role of activation functions is crucial in CNN to segregate the useful and not much helpful information from the neurons. Non-linearity is introduced into the network using activation functions. Over the choices of activation functions like logistic function, $\tanh(x)$ function, arctan function, and more, these functions lead to vanishing the gradients [4]. Since the input to the neurons is a range of 0, whereas the gradients are significant values. Rectified Linear Unit (ReLU) is used to overcome this problem.

$$\text{ReLU}(x) = \max(x, 0) \quad (7)$$

4 Optimizer

4.1 Stochastic Gradient Descent (SGD)

SGD optimizers update the parameter for every training example and its label [30]. Let x_i is the input and its labels y_i . $J(\theta; x_i; y_i)$ in the equation represents the objective function. Gradient Descent is a way to minimize the cost. $J(\theta; x_i; y_i)$ is parameterized by the model's parameters $\theta \in \mathbb{R}^d$. θ represents the weights, that gets updated for every iteration. η represents the learning rate that reflects the size of the steps that take to reach a local minimum.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x_i; y_i) \quad (8)$$

Batch Gradient Descent (BGD) performs excess calculations for massive datasets, as it recomputes gradients for similar examples before every parameter gets updated. SGD gets rid of this repetition by performing each update at a time. It is in this manner typically a lot quicker and can likewise be utilized to learn online.

SGD performs updates frequently with a large difference that causes the target function to change vigorously as in image. SGD enables a better local minimum than BGD.

4.1.1 Adadelata

Adagrad algorithm adapts the lower learning rate of the parameters with lower learning rates where features occur frequently and the higher learning rate for the parameters with infrequent features. Adelta is an extension of Adagrad that optimizes the learning rate. Rather than aggregating all past squared gradients, it limits the window of collected past gradients to some fixed size w

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma)g_t^2 \quad (9)$$

where $E[g^2]_t$ is moving average, t is the time step, and γ is the momentum. Here, γ value is set to 0.9.

4.2 Fully Connected Layers

All the nodes in one layer are connected to the output of the other layers is the Fully Connected (FC) layer. The FC layer outputs the class probabilities, where each class is assigned a probability number. All probabilities must sum to 1. Softmax functions produce the probability values for all the categories. All the outputs of a fully connected layer are probabilities. Applying softmax squashes the real value numbers into probabilities that sum to 1. Softmax comprises the output range of (0,1)

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)} \quad (10)$$

4.3 Need of ReLU (Non-linearity) and Modification in LeNet 5

Non-linear functions are required for universal approximation. Adding more layers into the neural network doesn't increase the approximation power. Nonlinearity in the decision function increases with the ReLU activation function. As per the universal approximation algorithm, using only linear activation function in the Neural Networks is less valid. When more layers are added into the network, ReLU is faster to train than CNN with $\tanh(x)$. The reason to make the network train slower is—Gradient Descent based optimizers are used for weight optimization and to achieve the local minima. Use of $\tanh(x)$ may lead to vanish the Gradient Descent of the network. Whereas in the $\tanh(x)$, x is the independent variable and it has the problem of reaching $\pm\infty$ and $\tanh(x)$ derivative goes to 0. The gradients become smaller and smaller which means the network cannot be trained anymore, whereas in ReLU, the derivative is a constant if $x > 0$. The relationship between the Network and input is non-linear with the use of non-linear activation function, which makes trained networks take complex decisions. In a regular LeNet 5 the activation function used is $\tanh(x)$ which is linear in nature, where a modification in LeNet 5 by updating Linear function to ReLU a non-linear function gives a better classification accuracy and achieved max local minima (discussed in the results session). in the below Table 6.

5 Experiment Results and Discussion

5.1 Experimental Data

5.1.1 Dataset

The data contains handwritten samples of Devanagari numerals and vowels (i.e., 10 numerals and 13 consonants). Thus, the dataset includes a total of 23 different Devanagari characters, as shown in Tables 1 and 2. The data is collected on a regular A4 sheet which was distributed to the subjects. The subjects wrote numerals and vowels in Devanagari Scripts. Targeted subjects are from schools and colleges. These sheets were scanned at 300 dpi using Epson DS – 150 is shown in Figs. 1, and 2. The numerals and vowels were collected from 2400 and 1400 subjects of different age groups (Figs. 3, 4, Tables 3, 4).

Characters are extracted from the scanned images using the segmentation algorithm mentioned in Sect. 2. Further, segmented data is pre-processed. This characters are manually segregated since noise in the scanned images are also obtained as characters. All the images were resized to 28×28 pixels, verified manually and converted into black and white. Where background was converted to black and character was converted to white, as shown in Tables 1 and 2. These are stored in a publicly accessible location. By removing the occluded images and scribbles, the final data set contains a total of 38,750 digitized images where 22,500 Devanagari Numerals (2250 each) and 16,250 Vowels (1250 each). Extracted images of Devanagari characters were arranged into different folders—a total of 23 folders where each folder for each character (Figs. 5, 6).

Table 1 Devanagari Numerals Representation











0	1	2	3	4	5	6	7	8	9
									
10	1	2	3	4	5	6	7	8	9

Table 2 Devanagari Vowels Representation













a	aa	e	ee	u	uu	ru	ye	i	o	au	am	aha
												
11	12	13	14	15	16	17	18	19	20	21	22	23

Table 3 Devanagari Characters that looks similar
















Vowels (pronunciation)	a	aa	u	uu	o	au	am	aha
Vowels Sample Image								
Class Number	11	12	15	16	20	21	22	23

Table 4 Devanagari Numerals that looks similar

Numerals	1	2	3	5	6	7	8
Numerals Sample Image							
Class Number	1	2	3	5	6	7	8

The success rate of research on the recognition of handwritten English characters is high when compared to the Indian script like Devanagari. The state-of-the-art techniques in deep learning are efficient in automatic recognition of Devanagari handwritten characters, but this requires large samples of data with labels. This data will fill that data-gap for Devanagari numerals and vowels. This data is publicly available at <https://data.mendeley.com/datasets/pxrnvp4yy8/1> (Figs. 7, 8).

In this research, CNN, MLCNN, and ACNN is used for implementing HDCR. Dataset used in this work is collected from 3,800 subjects of different ages from schools, colleges, and other professions. The number of images filtered after pre-processing is 38,750. The data is split into different ratios of 80:20, 70:30, and 60:40 for training and testing, as mentioned in Table 10 (Figs. 9, 10).

5.2 Experimental Settings

In this research, the input image characters are resized to 28×28 , following the standards of the MNIST dataset. If the size of the character has increased, the accuracy may also increase, which is directly proportional to the computational cost. So, to avoid more computational cost, the size is fixed to 28×28 . For training the network, the batch size is set to 32. Both the numerals and alphabets are trained separately as per Table 8.

5.2.1 For CNN

Along with the above settings, SGD is used to train the model at the learning rate of 0.01. In the first layer, the network is initialized with 36 kernels and increased to 64 kernels. Dropouts are added to avoid overfitting with the ratio of 25%, and 50% of neurons are dropped before the final layer. The architecture of CNN is as mentioned in the Table 5.

5.2.2 For MLCNN

Convolution kernels used in the MLCNN network is 6 and 16 in different layers, which are very useful. Strides are used with values 1 and 2 to control the size of the convolution layer in subsampling. The padding of [1 1 1 1] is implemented in the second and fourth layers. In this NN, SGD is used with a learning rate of 0.01. It is prepared to update the value of the learning rate if the error rate is compromised. The architecture of MLCNN is as mentioned in the Table 6.

Table 5 CNN Architecture

	Layer (type)	Image input	Param	No. of params	Activation function
1	'conv1'	Convolution	32 kernel 3×3 and input shape 28×28	320	ReLU
2	'pool1'	Max pooling	2×2 Avg Pooling, input shape $26 \times 26 \times 32$	–	–
2	'conv2'	Convolution	64 kernel 3×3 and input shape $13 \times 13 \times 32$	18,496	ReLU
3	'pool1'	Max pooling	2×2 Avg Pooling, input shape $11 \times 11 \times 64$	–	–
4	flatten	Flatten	1600, input shape $5 \times 5 \times 64$	–	–
5	'fc1'	Fully connected	Fully connected with dense 128	1,179,776	ReLU
6	'drop2'	Dropout	50% dropout	–	–
7	'fc2'	Fully connected	Fully connected with dense 10	1290	Softmax
Total param: 1,199,882				Non-trainable params: 0	

Table 6 Modified Lenet CNN Architecture

	Layer (type)	Image input	Param	No. of params	Activation function
1	'conv1'	Convolution	6 kernel 5×5 and input shape 28×28	156	ReLU
2	'pool1'	Avg pooling	2×2 Avg Pooling, padding [1 1 1 1], Stride 2, input shape $27 \times 27 \times 6$		
3	'conv2'	Convolution	16 kernel 5×5 , Stride 2, input shape $23 \times 23 \times 6$	2416	ReLU
4	'pool2'	Average pooling	2×2 Avg pooling, padding [1 1 1 1], Stride 1, input shape $11 \times 11 \times 16$		
5	'fc1'	Fully connected	Fully connected with dense 120, input shape $5 \times 5 \times 16$	232,440	ReLU
6	'fc2'	Fully connected	Fully connected with dense 84	10,164	ReLU
7	'fc3'	Fully connected	Fully connected with dense 13 / 10	1105	Softmax
Total param: 246,281 Trainable params: 246,281				Non-trainable params: 0	

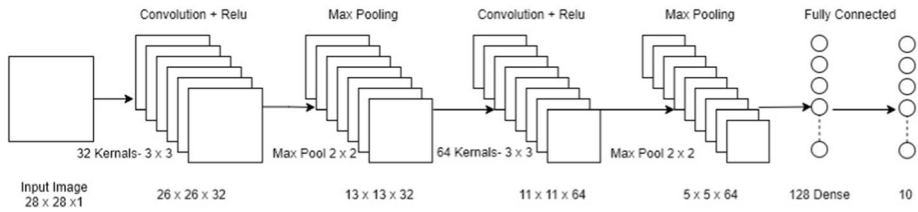


Fig. 3 CNN Architecture

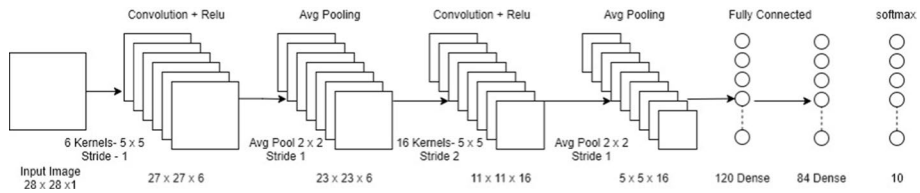


Fig. 4 Modified LeNet CNN Architecture

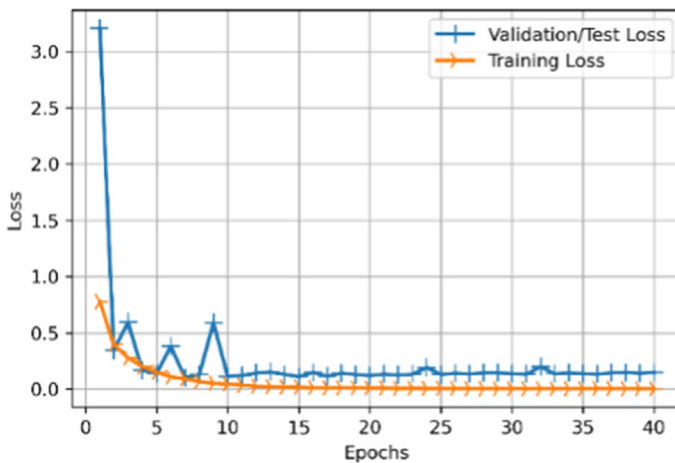


Fig. 5 Loss for 40 Epochs for ACNN for 80:20 Vowels Dataset

5.2.3 For ACNN

ACNN has its first layers with 28 kernels that are regularized with the L2 regularization technique. L2 regularization helps to avoid the overfitting of data from the first layer itself. Batch normalizations are added to all the convolution layers with activation function ReLU that introduces non-linearity to the data. Kernels that are used in each layer are 56, 112, 224, 448 with the window size of (3,3) (5,5) (7,7) (3,3), and (3,3), respectively. Since there are essential parameters in Alexnet CNN – to avoid further overfitting, dropout is added at the ratio of 50% for the first two fully connected layers. The final output layer is softmax. The architecture of MLCNN is as mentioned in the Table 7.

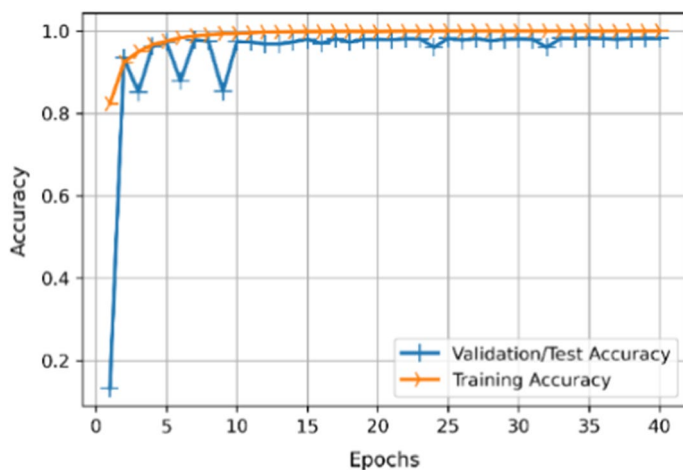


Fig. 6 Accuracy for 40 Epochs for ACNN for 80:20 Vowels Dataset

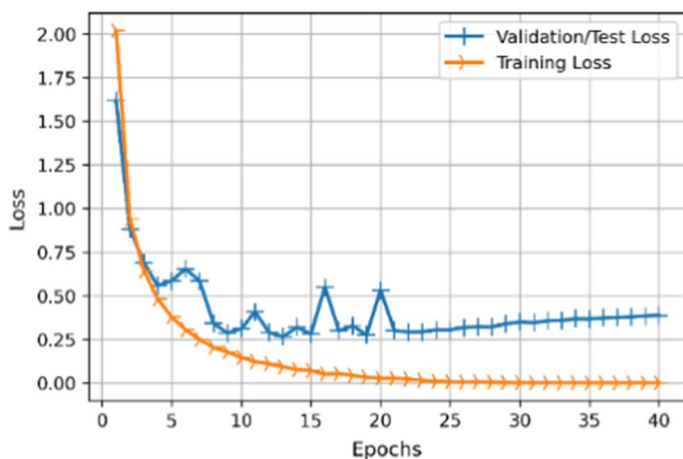


Fig. 7 Loss for 40 Epochs for MLCNN for 80:20 Vowels Dataset

5.3 Results Comparison and Training strategy of CNN, MLCNN and ACNN

Three different CNN architectures are designed before performing classification and trained as per the ratios in Table 8. Table 11 shows the training accuracies achieved by CNN, MLCNN, and ACNN for alphabets and numerals (Fig. 11, Tables 9, 10).

A simple CNN architecture is used in this research for both alphabets and numerals. Alphabets achieved an accuracy of 96.88% of efficiency for 70:30 split ratio at 50 epochs and 96.29%, 96.00% for 80:20 and 60:40 as shown in the Fig. 12 split ratio. ACNN architecture obtained an accuracy of 99.99% is collected for 40 epochs training of alphabets at 80:20 ratio split as shown in the Fig. 8. The other accuracies obtained

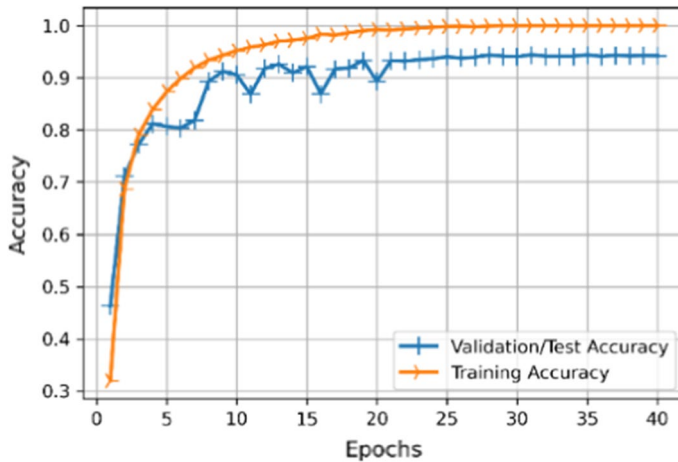


Fig. 8 Accuracy for 40 Epochs for MLCNN for 80:20 Vowels Dataset

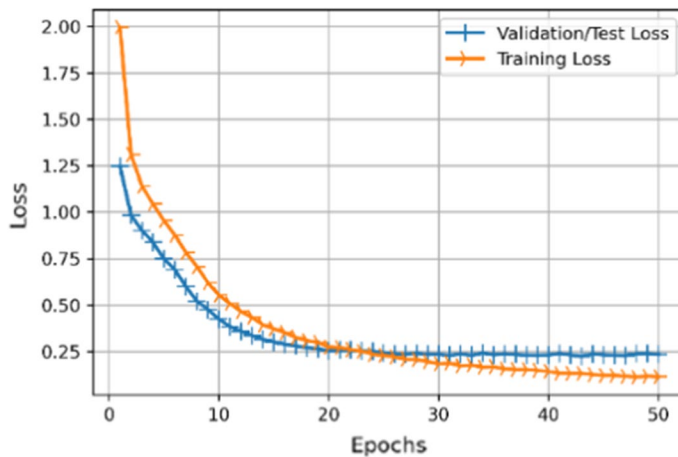


Fig. 9 Loss for 50 Epochs for CNN for 60:40 Vowels Dataset

at 70:30 and 60:40 is 99.98% and 99.97%, respectively. The training of alphabets is stopped at 40 epochs; the reason for not recording the 50 epochs training is—loss values beyond 40 epochs are increasing and crossing the local minima in the opposite direction. A similar problem was raised for MLCNN. The training strategy used in MLCNN is, where SGD is using as the optimizer, the learning rate is updated to 0.001 from 0.01. The advantage of using SGD over the Adadelta optimizer is tuning the learning rate. By slowing down the learning rate, the accuracy is compromised. In MLCNN, at 33rd epoch loss is increasing so, the learning rate is tuned to 0.001. Later, it is observed that the cost value is not reversed when it is trained for 50 epochs. When the network is trained for 40 epochs as shown in the Fig. 10, the accuracy obtained is 99.98%, 99.98%,

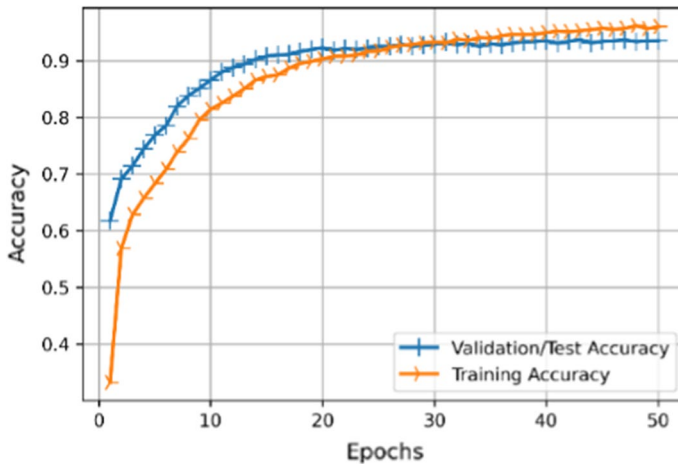


Fig. 10 Accuracy for 50 Epochs for CNN for 60:40 Vowels Dataset

and 99.97% for 80:20, 70:30 and 60:40 at 33rd, 35th, and 35th epoch simultaneously. The problem of overfitting was raised for alphabets, because of 13 classes in HDCR of alphabets. Eight categories of characters look similar to minor changes, as shown in Table 3.

{1,2}, {3, 6}, and {5, 7, 8} sets of numbers in Devanagari have structural similarity as shown in Table 4. CNN architecture obtained training accuracy of 97.22%, 96.86%, and 96.77% at different splits. ACNN obtained top accuracy of 99.87% for 80:20 ratio at 40 epochs as shown in the Fig. 12 and Fig. 13. MLCNN has shown promising results with the least loss for both training and testing, as shown in Table 12. It obtained an accuracy of 99.84%, 99.83%, and 99.82% at different splits and epochs (Fig. 14).

A trained model efficiency can be calculated not only based on training data accuracy. But, based on accuracy, it gives, when it encounters unseen data. Table 12 indicates the accuracy achieved by the sensed data by the neural network model. Overall, the training accuracies of all the three models are similar when it is compared to testing accuracy/ validation accuracy or accuracy on the unseen dataset; there are some differences in the values. ACNN for alphabets obtained an accuracy of 98.25% accuracy for 80:20 split at 40 epochs. The regularization techniques L2 and dropouts are used in this ACNN helped to give better accuracy. 98.25% and 98.32% for 70:30 and 60:40 split of data. Whereas, numerals are considered 97.22% of accuracy is achieved at 70:30 for 40 epochs. The loss calculated over epochs for vowels dataset are shown in the Figs. 7, 9 and 11 for ACNN, MLCNN and CNN respectively. The loss calculated for MLCNN and CNN for numerals dataset is shown in the Figs. 11 and 14.

Each Deep Learning Network is trained and tested for 12 number of times for alphabets data and 12 number of times for numbers data. This process is categorized into 4,

Table 7 Alexnet CNN Architecture

	Layer (type)	Image input	Param	No. of params
1	'conv1'	Convolution	28 kernel 3×3 and input shape 28×28	280
2	'norm1'	Batch normalization	Batch normalization	112
3	'activation1'	ReLU	ReLU	—
4	'pool1'	Max pooling	2×2 max pooling	—
5	'conv2'	Convolution	56 kernel 5×5 and padding [1 1 1 1]	39,256
6	'norm2'	Batch normalization	Batch normalization	224
7	'activation2'	ReLU	ReLU	—
8	'pool2'	Max pooling	2×2 max pooling	—
9	'zeropadding3'	Zero padding	Zero Padding [1 1 1 1]	—
10	'conv3'	Convolution	112 kernel 7×7 and padding [1 1 1 1]	307,440
11	'norm3'	Batch normalization	Batch normalization	448
12	'activation3'	ReLU	ReLU	—
13	'pool3'	Max pooling	2×2 max pooling	—
14	'zeropadding4'	Zero padding	Zero Padding [1 1 1 1]	—
15	'conv4'	Convolution	224 kernel 3×3 and padding [1 1 1 1]	226,016
16	'norm4'	Batch normalization	Batch normalization	896
17	'activation4'	ReLU	ReLU	—
18	'zeropadding5'	Zero padding	Zero padding [1 1 1 1]	—
19	'conv5'	Convolution	448 kernel 3×3 and padding [1 1 1 1]	903,616
20	'norm5'	Batch normalization	Batch normalization	1792
21	'activation5'	ReLU	ReLU	—
22	'pool5'	Max pooling	2×2 max pooling	—
23	'fc6'	Fully connected	Fully connected Layer with dense 786	5,620,496
24	'norm6'	Batch normalization	Batch normalization	3136
25	'activation6'	ReLU	ReLU	—
26	'drop6'	Dropout	50% dropout	—
27	'fc7'	Fully connected	Fully connected layer with dense 1204	945,140

Table 7 (continued)

	Layer (type)	Image input	Param	No. of params
28	'norm7'	Batch normalization	Batch normalization	4816
29	'activation7'	ReLU	ReLU	–
30	'drop7'	Dropout	50% dropout	–
31	'fc8'	Fully connected	Fully connected layer with dense 10	15,665
32	'norm8'	Batch normalization	Batch normalization	52
33	'prob'	Softmax	Softmax	–
34	'Output'	Classification output	Crossentropy with Adadelta optimizer	–
Total param:		Trainable params: 8,063,647		
		Non-trainable params: 5738		
		8,069,385		

Table 8 Data split for training and testing

Training: testing	Numerals		Alphabets	
	Training	Testing	Training	Testing
80:20	18,000	4500	13,000	3250
70:30	15,750	6750	11,375	4875
60:40	13,500	9000	9750	6500

Table 9 Accuracy on training data

Training Accuracies for Alphabets (13 Classes)				Training Accuracies for Numerals (10 Classes)			
Train: Test	ACNN	MLCNN	CNN	Train: Test	ACNN	MLCNN	CNN
20 Epochs				20 Epochs			
80:20	99.92	99.10	91.95	80:20	99.69	99.82	95.54
70:30	99.89	99.31	90.63	70:30	99.65	99.82	95.37
60:40	99.96	97.81	91.37	60:40	99.61	99.81	94.36
30 Epochs				30 Epochs			
80:20	99.99	99.98	94.01	80:20	99.79	99.82	96.10
70:30	97.91	99.96	93.83	70:30	99.86	99.84	96.01
60:40	99.22	99.91	93.22	60:40	99.84	99.83	95.42
40 Epochs				40 Epochs			
80:20(39)	99.99	99.98	95.32	80:20	99.87	99.82	96.66
70:30	99.98	99.98	95.50	70:30	99.85	99.81	96.56
60:40	99.97	99.97	94.91	60:40	99.87	99.81	96.04
50 Epochs				50 Epochs			
80:20	–	95.32	96.29	80:20	99.88	99.83	97.22
70:30	–	94.28	96.68	70:30	99.83	99.81	96.86
60:40	–	92.53	96.00	60:40	99.81	99.81	96.77

i.e., 20 epochs, 30 epochs, 40 epochs, and 50 epochs. Every category is further trained and test based on the split of data into different ratios, i.e., 80:20, 70:30, 60:40. The proposed deep learning architectures (CNN, MLCNN, and ACNN) are developed and tested on the dataset for 12 times. The dataset is tested for 72 times. For every instance, four parameters are taken to check the model stability. Training and testing accuracy and loss values are collected. Among the 72 different sets of values, Table 11 contains the optimal values. A study measures it, maximum accuracy of both training and testing, along with the minimum costs of training and testing loss.

Table 10 Accuracies calculated on the unseen data

Testing Accuracies for Alphabets (13 Classes)				Testing Accuracies for Numerals (10 Classes)			
Train: Test	ACNN	MLCNN	CNN	Train: Test	ACNN	MLCNN	CNN
20 Epochs				20 Epochs			
80:20	98.03	91.41	92.46	80:20	97.07	96.56	96.0
70:30	97.74	88.86	91.85	70:30	96.17	96.49	95.44
60:40	97.40	88.40	91.88	60:40	97.30	96.57	95.79
30 Epochs				30 Epochs			
80:20	98.25	91.56	93.85	80:20	97.20	96.64	96.24
70:30	97.91	91.87	93.05	70:30	96.74	96.41	96.04
60:40	98.26	90.88	92.08	60:40	97.27	96.41	95.86
40 Epochs				40 Epochs			
80:20(39)	98.25	94.09	93.54	80:20	97.16	96.71	96.18
70:30	98.24	91.73	93.62	70:30	97.22	96.40	96.21
60:40	98.32	91.11	93.05	60:40	97.27	96.49	96.20
50 Epochs				50 Epochs			
80:20	–	88.86	94.31	80:20	97.27	96.58	96.44
70:30	–	87.53	94.21	70:30	97.28	96.55	96.44
60:40	–	85.83	93.52	60:40	97.26	96.46	96.28

5.4 Comparison of Different Methods HDCR

To show the pre-eminence of the proposed architecture and dataset efficiency, we compare the performances of different models from the recent review articles [2, 6, 8]. The parameter considered for comparison is the size of the dataset, because, lower the size of the dataset leads to cover fewer variant styles and may achieve higher accuracies but when the dataset size and more variants of writing styles are introduced in the dataset may lead to a drop of accuracies. In the Table 12, the highest accuracy achieved is 99% in [32], whereas the size of the dataset is 5424. Similarly, the dataset of sized 10,000 [31], 25,000 [18], 22,556 [23], and 32,413 [32], got accuracy of 97.18%, 90.11%, 93.4% and 61.8% respectively. It is observed that the accuracy is reduced when more variants of data or data size increase. In this article, the dataset size of 22,500 images could able to achieve 99.9% training data and 98.25% for unseen datasets for numbers. For the dataset size of 16,250 images, could able to achieve 99.87% for training data and 97.16 for unseen data. our model has significant improvement.

Table 11 Top Accuracies

ACNN for Alphabets (13 Classes)				ACNN for Numerals (10 Classes)					
40 Epochs				40 Epochs					
Train: Test	Training Accuracy	Testing Accuracy	Train Loss	Test Loss	Train: Test	Training Accuracy	Testing Accuracy	Train Loss	Test Loss
80:20	99.99	98.25	0.001	0.14	80:20	99.87	97.16	0.003	0.20
70:30	99.98	98.24	0.001	0.13	70:30	99.85	97.22	0.004	0.171
60:40	99.98	98.32	0.001	0.11	60:40	99.87	97.27	0.003	0.17
MLCNN for Alphabets (13 Classes)				MLCNN for Numerals (10 Classes)					
40 Epochs				30 Epochs					
Train: Test	Training Accuracy	Testing Accuracy	Train Loss	Test Loss	Train: Test	Training Accuracy	Testing Accuracy	Train Loss	Test Loss
80:20 (33)	99.98	94.09	0.001	0.34	80:20	99.82	96.64	0.002	0.21
70:30 (35)	99.98	91.73	0.0010	0.50	70:30	99.84	96.41	0.002	0.21
60:40(35)	99.97	91.11	0.001	0.50	60:40	99.83	96.41	0.002	0.20
CNN for Alphabets (13 Classes)				CNN for Numerals (10 Classes)					
50 Epochs				50 Epochs					
Train: Test	Training Accuracy	Testing Accuracy	Train Loss	Test Loss	Train: Test	Training Accuracy	Testing Accuracy	Train Loss	Test Loss
80:20	96.29	94.31	0.10	0.21	80:20	97.22	96.44	0.07	0.12
70:30	96.68	94.21	0.09	0.23	70:30	96.86	96.44	0.08	0.12
60:40	96.00	93.52	0.11	0.23	60:40	96.77	96.28	0.09	0.12

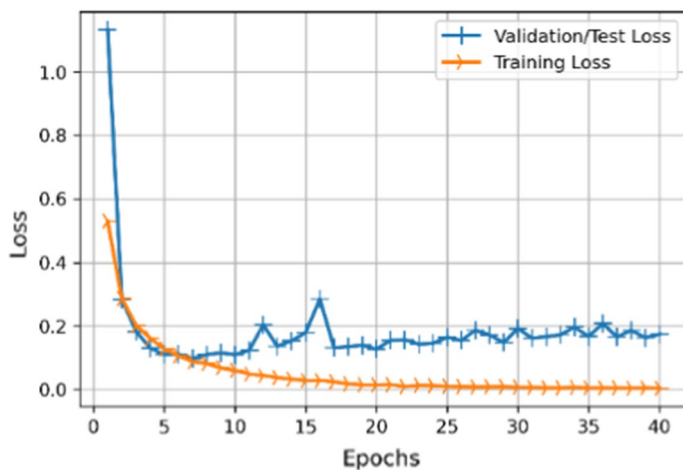


Fig. 11 Loss for 50 Epochs for MLCNN for 80:20 Numerals Dataset

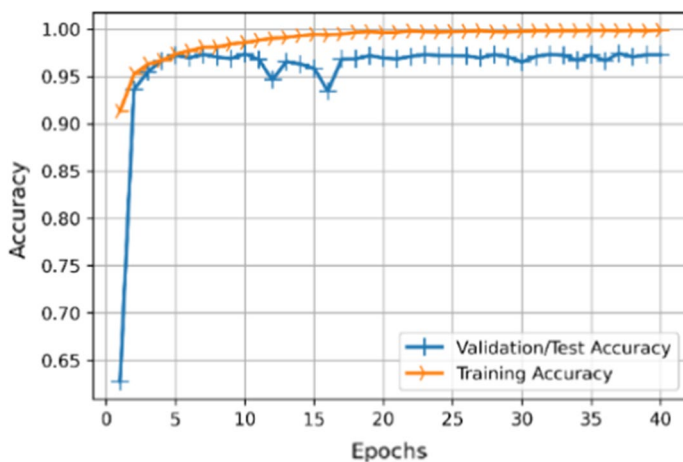


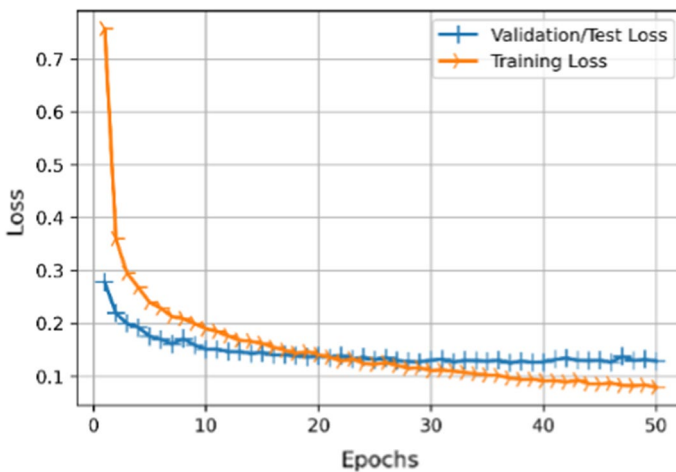
Fig. 12 Accuracy for 50 Epochs for MLCNN for 80:20 Numerals Dataset

6 Conclusion

In this paper, we provide a dataset of 38,750 images of Devanagari numerals and vowels collected from 2400 subjects of different age groups. We also developed a segmentation algorithm suitable for the dataset. A modified CNN is proposed to solve the HDCR

Table 12 Different methods and accuracy of HDCR

Method	Feature	Accuracy (%)	Dataset
Tree based classifier and Template matching based approach [31]	Structural features	97.18	10,000
KNN and neural network [32]	Gradient features	61.8	32,413
MLP and SVM classifier [18]	Chain code, Kirsch Directional edges, Gradient, distance transform	90.116 by SVM and 87.56 by MLP	25,000
Neural network classifier [23]	Combination of structural and statistical features	93.4	22,556
MQDF [32]	Directional features	99.0	5424
MLCR [34]	Geometric distance based	98.1	3100
CNN [35]	Convolution	95	4282
KNN [36]	Hu's seven variants and zernike moment features	89	2600
MLCNN	Modified LeNet convolutional neural network	99.9	38,750

**Fig. 13** Loss for 50 Epochs for CNN for 80:20 Numerals Dataset

problem. The dataset developed followed the standards of MNIST and served as a benchmark dataset for Devanagari numerals and vowels. Experiments are conducted by using three different deep learning architectures, 1. CNN, 2. MLCNN, and 3. ACNN. It achieved an error rate of 0.001% and 99.98% accuracy. Through a series of experiments, we have shown that MLCNN and ACNN are highly efficient (99.9% and 99.8%). ACNN's computational cost is high when compared to MLCNN but can overcome by GPU based systems.

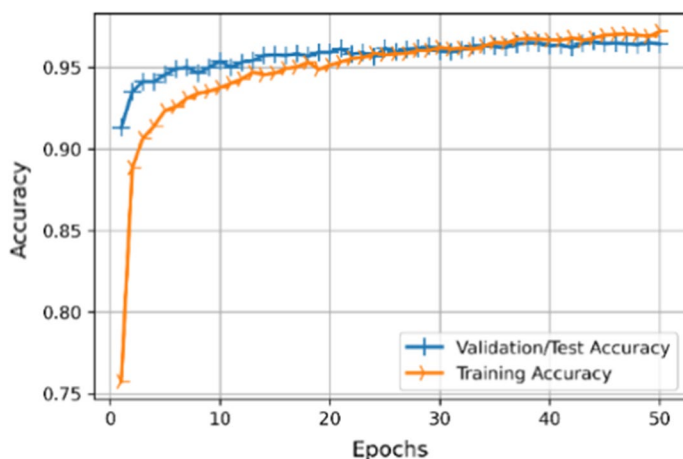


Fig. 14 Loss for 50 Epochs for CNN for 80:20 Numerals Dataset

We implemented a CNN by modifying LeNet 5 CNN, which reduces the time significantly, where these models can be used in online HDCR. The performance of ACNN can be further improved through the fine-tuning of its structure and its parameters. In future work, we plan to develop a combined model for all the Devanagari characters of more than 50 classes that include numeral (10), vowels (13), and consonant (33).

Authors' contributions Duddela Sai Prashanth: Conceptualization, Methodology, Software, Visualization R Vasanth Kumar Mehta: Data curation, Writing—original draft, Data Analysis, Investigation. Kadiyala Ramana: Software, Validation, Editing. Vidya Charan Bhaskar: Supervision, Writing—review & editing.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Availability of data and material The datasets generated during and/or analysed during the current study are available in the Mendeley repository, <https://data.mendeley.com/datasets/pxrnvp4yy8/1>

Code availability The code that supports the findings of this study are available on request from the corresponding author. The code is not publicly available due to containing information that could compromise the privacy of research participants.

Declarations

Conflict of interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Sahare, P., & Dhok, S. B. (2019). Robust character segmentation and recognition schemes for multilingual indian document Images. *IETE Technical Review*, 36(2), 209–222.
2. Parekh, K. A., Goswami, M. M., & Mitra, S. K. (2020). Handwritten numeral recognition using polar histogram of low-level stroke features. In *Proceedings of 3rd international conference on computer vision and image processing* (pp. 169–181). Springer.

3. Bansal, V., & Sinha, R. M. K. (2002). Segmentation of touching and fused Devanagari characters. *Pattern recognition*, 35(4), 875–893.
4. Li, Z., Teng, N., Jin, M., & Lu, H. (2018). Building efficient CNN architecture for offline handwritten Chinese character recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 21(4), 233–240.
5. Kaur, S., & Sagar, B. B. (2019). Brahmi character recognition based on SVM (support vector machine) classifier using image gradient features. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(8), 1365–1381.
6. Setlur, S. (2009). Guide to OCR for indic scripts. In V. Govindaraju (Ed.). Springer.
7. Bathla, A. K., Gupta, S. K., & Jindal, M. K. (2016, March). Challenges in recognition of Devanagari Scripts due to segmentation of handwritten text. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 2711–2715). IEEE.
8. <https://data.mendeley.com/datasets/pxrnvp4yy8/1>
9. Bharath, A., & Madhvanath, S. (2011). HMM-based lexicon-driven and lexicon-free word recognition for online handwritten Indic scripts. *IEEE transactions on pattern analysis and machine intelligence*, 34(4), 670–682.
10. Soora, N. R., & Deshpande, P. S. (2018). Review of feature extraction techniques for character recognition. *IETE Journal of Research*, 64(2), 280–295.
11. Sharma, A. K., Adhyaru, D. M., & Zaveri, T. H. (2020). A survey on Devanagari character recognition. In *Smart systems and IoT: Innovations in computing* (pp. 429–437). Springer.
12. Goswami, M. M., Prajapati, H. B., & Dabhi, V. K. (2011). Classification of printed Gujarati characters using SOM based k-Nearest Neighbor Classifier. In *2011 International conference on image information processing* (pp. 1–5). IEEE.
13. Sinha, R. M. K., & Mahabala, H. N. (1979). Machine recognition of Devanagari script. *IEEE Transactions on Systems, Man and Cybernetics*, 9(8), 435–441.
14. Jayanthi, K., Suzuki, A., Kanai, H., Kawazoe, Y., Kimura, M., & Kido, K. (1989, November). Devanagari character recognition using structure analysis. In *Fourth IEEE Region 10 international conference TENCON* (pp. 363–366). IEEE.
15. Bansal, V., & Sinha, M. K. (2001, September). A complete OCR for printed Hindi text in Devanagari script. In *Proceedings of Sixth International Conference on Document Analysis and Recognition* (pp. 0800–0800). IEEE Computer Society.
16. Sharma, N., Pal, U., Kimura, F., & Pal, S. (2006). Recognition of offline handwritten devnagari characters using quadratic classifier. In *Computer vision, graphics and image processing* (pp. 805–816). Springer.
17. Arora, S., Bhattacharjee, D., Nasipuri, M., Basu, D. K., Kundu, M., & Malik, L. (2009, December). Study of different features on handwritten Devnagari character. In *2009 Second international conference on emerging trends in engineering & technology* (pp. 929–933). IEEE.
18. Kumar, S. (2009). Performance comparison of features on Devanagari hand-printed dataset. *International Journal of Recent Trends in Engineering*, 1(2), 33.
19. Arora, S., Bhattacharjee, D., Nasipuri, M., Basu, D. K., & Kundu, M. (2010). Recognition of non-compound handwritten devnagari characters using a combination of mlp and minimum edit distance. arXiv preprint arXiv:1006.5908.
20. Pal, U., Chanda, S., Wakabayashi, T., & Kimura, F. (2008). Accuracy improvement of Devnagari character recognition combining SVM and MQDF. In *Proceedings of 11th International Conference on Frontiers Handwrit. Recognition* (pp. 367–372).
21. Kale, K. V., Deshmukh, P. D., Chavan, S. V., Kazi, M. M., & Rode, Y. S. (2013). Zernike moment feature extraction for handwritten Devanagari compound character recognition. In *2013 Science and information conference* (pp. 459–466). IEEE.
22. Arora, S., Bhattacharjee, D., Nasipuri, M., & Malik, L. (2007). A two stage classification approach for handwritten Devnagari characters. In *International conference on computational intelligence and multimedia applications (ICCIMA 2007)* (Vol. 2, pp. 399–403). IEEE.
23. Hanmandlu, M., Murthy, O. R., & Madasu, V. K. (2007, December). Fuzzy model based recognition of handwritten Hindi characters. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)* (pp. 454–461). IEEE.
24. Aneja, N., & Aneja, S. (2019, July). Transfer learning using CNN for handwritten Devanagari character recognition. In *2019 1st International Conference on Advances in Information Technology (ICAIT)* (pp. 293–296). IEEE.

25. Arora, S., Bhattacharjee, D., Nasipuri, M., Malik, L., Kundu, M., & Basu, D. K. (2010). Performance comparison of SVM and ANN for handwritten devnagari character recognition. *arXiv preprint arXiv:1006.5902*
26. Lu, S., Lu, Z., & Zhang, Y. D. (2019). Pathological brain detection based on AlexNet and transfer learning. *Journal of computational science*, 30, 41–47.
27. Shima, Y., Nakashima, Y., & Yasuda, M. (2018, April). Handwritten Digits Recognition by Using CNN Alex-Net Pre-trained for Large-scale Object Image Dataset. In *Proceedings of the 3rd international conference on multimedia systems and signal processing* (pp. 36–40).
28. Kumar, M., Jindal, M. K., Sharma, R. K., & Jindal, S. R. (2019). Character and numeral recognition for non-Indic and Indic scripts: A survey. *Artificial Intelligence Review*, 52(4), 2235–2261.
29. Niu, X. X., & Suen, C. Y. (2012). A novel hybrid CNN–SVM classifier for recognizing handwritten digits. *Pattern Recognition*, 45(4), 1318–1325.
30. Jiang, B., He, J., Yang, S., Fu, H., Li, T., Song, H., & He, D. (2019). Fusion of machine vision technology and AlexNet-CNNs deep learning network for the detection of postharvest apple pesticide residues. *Artificial Intelligence in Agriculture*, 1, 1–8.
31. Chaudhuri, B. B., & Pal, U. (1997, August). An OCR system to read two Indian language scripts: Bangla and Devnagari (Hindi). In *Proceedings of the fourth international conference on document analysis and recognition* (Vol. 2, pp. 1011–1015). IEEE.
32. Kompalli, S., Nayak, S., Setlur, S., & Govindaraju, V. (2005, August). Challenges in OCR of Devanagari documents. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)* (pp. 327–331). IEEE.
33. Pal, U., Sharma, N., Wakabayashi, T., & Kimura, F. (2007, September). Handwritten numeral recognition of six popular Indian scripts. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (Vol. 2, pp. 749–753). IEEE.
34. Soora, N. R., & Deshpande, P. S. (2017). Novel geometrical shape feature extraction techniques for multilingual character recognition. *IETE Technical Review*, 34(6), 612–621.
35. Prashanth, D. S., Mehta, R. V. K., & Sharma, N. (2020). Classification of Handwritten Devanagari number-an analysis of pattern recognition tool using neural network and CNN. *Procedia Computer Science*, 167, 2445–2457.
36. Prashanth, D. S., & Panini, C. N. (2017). KNN classification of Kannada Characters using Hu's Seven Variants and Zernike Moment. *Nagendra Panini Challa* on Aug, 10.
37. Latif, G., et al. (2018). Deep convolutional neural network for recognition of unified multi-language handwritten numerals. *2018 IEEE 2nd International workshop on Arabic and derived script analysis and recognition (ASAR)*. IEEE.
38. Chakraborty, B., et al. (2018). Does deeper network lead to better accuracy: a case study on handwritten Devanagari characters. *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE.
39. Rojatar, D. V., Chinchkhede, K. D., & Sarate, G. G. (2013). Design and analysis of LRTB feature based classifier applied to handwritten Devnagari characters: A neural network approach. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE,.
40. Ashokkumar, P., Siva Shankar, G., Gautam Srivastava, Praveen Kumar Reddy Maddikunta, & Thippa Reddy Gadekallu. 2021. "A Two-stage Text Feature Selection Algorithm for Improving Text Classification". *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 20, 3, Article 49 (July 2021), 19 pages. DOI:<https://doi.org/10.1145/3425781>
41. Dhanamjayulu, C., Nizhal, U. N., Kumar Reddy Maddikunta, P., Thippa Reddy Gadekallu, Celestine Iwendi, Chuliang Wei, & Qin Xin. (2021). Identification of malnutrition and prediction of BMI from facial images using real-time image processing and machine learning. *IET IMAGE PROCESSING*.
42. Gadekallu, T. R., Alazab, M., Kaluri, R., Reddy Maddikunta, P. K., Bhattacharya, S., Lakshman, K., & Parimala, M. (2021). Hand gesture classification using a novel CNN-crow search algorithm. *Complex & Intelligent Systems*, 7, 1–14.
43. Kothai, G., Poovammal, E., Dhiman, G., Ramana, K., Sharma, A., AlZain, M. A., & Masud, M. (2021). A new hybrid deep learning algorithm for prediction of wide traffic congestion in smart cities. *Wireless Communications and Mobile Computing*, 2021, 1.



Duddela Sai Prashanth, received M.Tech from Indian Institute of Information Technology, Allahabad and pursuing his PhD from SCSVMV University. He is serving as Assistant Professor from 2015 in Sahyadri College of Engineering & Management, Mangaluru. He is Co-Principal Investigator for the project sanctioned 30,00,000 INR by Vision Group of Science and Technology to establish Center of Excellence in Artificial Intelligent and Machine Learning.



Dr R Vasanth Kumar Mehta received M.S from BITS Pilani and the PhD Degree from SCSVMV University. He Joined SCSVMV University as Assistant Professor in the year 2001, where he is now Associate Professor. He is serving the University with the different responsibilities, as Board of Studies – Chairman, enabling off-campus internships and placements, Interaction with industries and academia in India and abroad for enhancing teaching, placements, industry grade projects and research. His Area of Interest lies in Data Mining, Distributed Computing Domain, Pattern Recognition. He is professional member of IEEE, CSI, ACM, IAENG. He is a member in Editorial Boards and Reviewer of reputed Journals.



Dr. Kadiyala Ramana is currently working as Associate Professor in Department of Artificial Intelligence & Data Science, AITS, Rajampet, India. He obtained his Bachelor of Technology degree in Information Technology from JNTUH, Hyderabad, India, M. Tech. from Satyabhama University, Chennai, India and completed his Ph.D. from SRM University, Chennai, India. He has 14 years of experience in teaching and research. He authored more than 30 international publications in Hindawi, Springer, IEEE etc. He is editorial board member and reviewer of several journals of international repute. His research interests Wireless Sensor Networks, Software-Defined Networking with Machine Learning and Data Analytics.



Dr. Vidhyacharan Bhaskar received the B.Sc. degree in Mathematics from the University of Madras, Chennai, India in 1992, M.E. degree in Electrical & Communication Engineering from the Indian Institute of Science, Bangalore in 1997, and the M.S.E. and Ph.D. degrees in Electrical Engineering from the University of Alabama in Huntsville in 2001 and 2002, respectively. During 2002–2003, he was a Post Doctoral fellow with the Communications research group at the University of Toronto, Canada, where he worked on the applications of space-time coding for wireless communication systems. During 2003–2006, he was an Associate Professor in the Department of Information Systems and Telecommunications at the University of Technology of Troyes, France. From Jan. 2007 to May 2014, he was a Full Professor in the Department of Electronics and Communication Engineering at S.R.M. University, Kattankulathur, India. Since 2015, he is a Professor in the Department of Electrical and Computer Engineering at San Francisco State University, San Francisco, California, USA. His research interests include MIMO wireless communications, signal processing, error control coding and queuing theory. He has published 130 Refereed Journal papers, presented around 72 Conference papers in various International Conferences over the past 20 years, published a book on “Adaptive Rate Coding for A-CDMA Systems” in Jan 2011, a book on “Higher-Order s-to-z mapping functions for digital filters,” in March 2013, and has also co-authored a book on MATLAB in 1998. He has 868 Google scholar citations. He has advised 50 Masters students, 11 Doctoral students, and 3 Senior design projects. He is an IEEE Senior member (SM-IEEE) and is a member of IET (M-IET, UK). He is a Fellow of Institute of Electronics and Telecommunication Engineers (F-IETE), and a Fellow of Institute of Engineers (F-IE), Kolkata, India. He is also a Life member of the Indian Society of Technical Education (LM-ISTE) and a member of the Indian Science Congress (M-ISC).

processing, error control coding and queuing theory. He has published 130 Refereed Journal papers, presented around 72 Conference papers in various International Conferences over the past 20 years, published a book on “Adaptive Rate Coding for A-CDMA Systems” in Jan 2011, a book on “Higher-Order s-to-z mapping functions for digital filters,” in March 2013, and has also co-authored a book on MATLAB in 1998. He has 868 Google scholar citations. He has advised 50 Masters students, 11 Doctoral students, and 3 Senior design projects. He is an IEEE Senior member (SM-IEEE) and is a member of IET (M-IET, UK). He is a Fellow of Institute of Electronics and Telecommunication Engineers (F-IETE), and a Fellow of Institute of Engineers (F-IE), Kolkata, India. He is also a Life member of the Indian Society of Technical Education (LM-ISTE) and a member of the Indian Science Congress (M-ISC).

Authors and Affiliations

Duddela Sai Prashanth^{1,2} · **R. Vasanth Kumar Mehta**¹ · **Kadiyala Ramana**³ · **Vidhyacharan Bhaskar**⁴ 

Duddela Sai Prashanth
dsaip13@gmail.com

R. Vasanth Kumar Mehta
vasanthmehta@gmail.com

Kadiyala Ramana
ramana.it01@gmail.com

¹ SCSVMV University, Kanchipuram, India

² Sahyadri College of Engineering and Management, Mangaluru, India

³ Department of Artificial Intelligence and Data Science, Annamacharya Institute of Technology and Sciences, Rajampet, Andhra Pradesh, India

⁴ Department of Electrical and Computer Engineering, San Francisco State University, San Francisco, CA 94132, USA