Rajpal and Garg
Journal of Engineering and Applied Science (2023) 70:81
https://doi.org/10.1186/s44147-023-00252-2

Journal of Engineering and Applied Science

RESEARCH Open Access

Ensemble of deep learning and machine learning approach for classification of handwritten Hindi numerals

Danveer Rajpal[1*] and Akhil Ranjan Garg[1]

*Correspondence:
danveer.rajpal@rediffmail.com

[1] Department of Electrical Engineering, M.B.M. University, Jodhpur 3420001, India

Abstract

Given the vast range of factors, including shape, size, skew, and orientation of handwritten numerals, their machine-based recognition is a difficult challenge for researchers in the pattern recognition field. Due to the abundance of curves and resembling shapes of the symbols, the recognition of Devnagari numerals can leverage the difficulty level of the recognition. The suggested low-classification-cost method for obtaining fine features from given numeral images used benchmark deep learning models, VGG-16Net, VGG-19Net, ResNet-50, and Inception-v3, to address these issues. Principal component analysis, a powerful dimensionality reduction method, was used to efficiently reduce the number of dimensions in the information that pre-trained deep convolutional neural network models provided. The method for improving recognition accuracy by fusing features was provided in the scheme. A machine learning algorithm: support vector machine was employed for the recognition task due to its capacity to distinguish between patterns belonging to distinct classes. The system was able to obtain a recognition accuracy of 99.72% and was effective in demonstrating the importance of ensemble machine learning and deep learning approaches.

Keywords: Deep convolutional neural networks, Deep learning, Dimensionality reduction, Feature optimization, Feature separation, Inception-v3, Machine learning, ResNet-50, Support vector machine, t-SNE, VGG-16Net, VGG-19Net

Introduction

Machine-based recognition of handwritten alphabets is one of the requirements of language-based automation. Intrinsic, unconditional diversity in writing styles, shapes, scales, skews, orientations, and deformations of handwritten alphabets are the main associated challenges. As a result of their massive populations not having embraced English as their first language, nations like India, China, Egypt, Saudi Arabia, and the United Arab Emirates are building automation systems in their own national tongues to benefit most of their populations. Many advancements have been reported for language-based automation systems related to English script due to its worldwide acceptance. Systems based on globally emerging languages like Hindi (Devnagari), Mandarin, Arabic,

Javanese, Urdu, and Persian require extra care. Efforts have been made in the present work for the Devanagari script. A set of Hindi numerals is shown in Fig. 1.

Several methods have been implemented so far for solving the proposed problem. Some benchmarking models are described as follows: Das et al. [1] received quad-tree longest-run and modular principal component analysis (PCA)-based features from numeral images and concatenated them. The classification was done with a one-versus-all support vector machine (SVM) classifier. Iamsa et al. [2] crafted a histogram of gradient (HOG) features from handwritten Hindi digits. The feedforward backpropagation neural network (FBNN) and extreme learning machine (ELM) were implemented as classification algorithms; the former was the top performer.

Khanduja et al. [3] created a hybrid of structural and statistical features that included intersection points, end points, loops, and pixel distributions. The feed-forward neural network was employed for the recognition of numerals. Singh et al. [4] examined the performance of five distinct classifiers: multilayer layer perceptron (MLP), Naïve Bayes (NB), logistic classifiers, random forest (RF), and SVM over local weighted run-length features received from numeral images. Acharya et al.'s [5] introduction of the deep convolutional neural networks (DCNN) model with a dropout function marked a turning point for Devanagari alphabet recognition algorithms. With the noble purpose of advancing relevant research, the authors have generated a benchmarking dataset of isolated handwritten Devanagari characters and made it freely accessible to the public. The effect of adding more layers to convolutional neural networks (CNN) on the recognition of Devanagari alphabets was studied by Chakraborty et al. [6]. A hybrid CNN and bidirectional long-short-term memory (BLSTM) model was also tried; however, it fell short of the performance of the standard CNN model. AlexNet, a pre-trained DCNN model, was used by Sonawane et al. [7] to present the transfer-learning method for identifying Devanagari characters. Aneja et al. [8] provided a thorough comparison analysis based on pre-trained DCNN models, including AlexNet, DenseNet-121, DenseNet-201, VGG-11, VGG-16, VGG-19, and Inception-V3, for the identification of Devanagari alphabets. Trivedi et al. [9] implemented a genetic algorithm and the L-BFGS optimization method to train CNN for addressing the concerns of getting stuck in local optima and the large number of iterations. Their evolutionary technique achieved a higher recognition rate

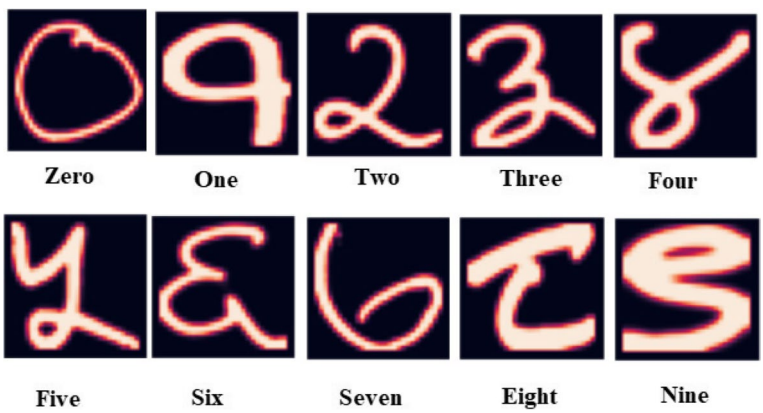

Fig. 1 Set of handwritten Hindi numeral

for handwritten Devanagari numerals. Kumar et al. [10] introduced a convolution autoencoder based on unsupervised learning to extract reduced-sized features from the augmented numeral images of Devanagari, English, and Bangla scripts. A deep convolutional network was employed for the final classification using these features. Chaurasia et al. [11] employed CNN as a feature extractor to receive salient features from handwritten numeral images of various Indian scripts. The authors employed an SVM classifier to avail the benefit of structural risk minimization. Sarkhel et al. [12] developed a state-of-the-art multicolumn, multi-scale CNN architecture for capturing important features from the images of handwritten characters related to several Indian scripts. A SVM classifier was employed for the classification task.

Some recent studies presented benchmark approaches to solving similar problems. Rakshit et al. [13] produced a comparative study of 11 different CNN models, namely, DenseNet-201, MobileNetV2, VGG-19, EfficientNetB0, NASNetMobile, Xception, Inception ResnetV2, ResNet50, EkushNet, InceptionV3, and ResNet152V2, in recognition of handwritten Bangla characters. ResNet152V2 was the top performer. Garg et al. [14] examined k-NN and SVM classifiers with linear, polynomial, and radial basis function (RBF) kernels in machine-based recognition of Gurumukhi characters. Peak extent and modified division point-based features were crafted for the purpose. In their later study [15], the authors presented a multifeature, multi-classifier approach for solving the problem of recognizing Gurumukhi script from degraded images. The authors employed zoning, diagonal, shadow, and peak extent-based features on k-NN, decision tree, and RF classifiers. Kathigi et al. [16] developed a skewed line segmentation technique to separate the individual Kannad characters. Steerable pyramid and discrete wavelet transforms were implemented to extract salient features. The classification was performed with LSTM using combined features. Narang et al. [17] employed CNN for feature extraction as well as for classification in the recognition of ancient characters in Devanagari script. Authors experimented with CNN architecture by varying counts of layers and filters, the size of stride and kernel, and activation functions in search of the best combination. To avoid manual feature engineering in the recognition of handwritten Urdu characters. Mushtaq et al. [18] developed a CNN model that outperformed the model based on handcrafted features. Robert Raj et al. [19] developed a recognition model for handling the problems of discontinuity, overlooping, and unnecessary portions presented in the structure of Tamil characters. The authors introduced a junction point elimination algorithm that outperformed conventional feature selection and pre-extraction algorithms. Deore et al. [20] finely tuned the popular deep convolutional neural network model VGG16 with advanced adaptive gradients to recognize handwritten Devanagari characters. Moudgil et al. [21] developed a convolution-based capsule network that captures spatial relationships among local features and reduces the vector length for effective classification of Devanagari characters. Guo et al. [22] proposed a solution for the recognition of similar-shaped Tai Le characters. The authors estimated the second- and third-level wavelet transforms for given character images and converted them into wavelet deep convolution features. Linear discriminant and principal component analysis were applied to limit the feature dimensionality. The classification model included six deep, variationally sparse Gaussian processes for efficient recognition. It

has been observed that deep learning techniques are replacing conventional feature extraction and classification techniques in this field in order to attain improved recognition accuracy [23].

It could be observed that the deep learning-based models achieved a significant recognition rate without the need for handcrafted features. The only concern is their large feature vectors, which may leverage the classification cost. Optimizing the size of the feature vector can lead to a low-classification-cost solution [24] in the following terms.

- *Training time*: A smaller feature vector typically implies fewer features that need to be processed and used to train a classifier. The computational complexity of training algorithms may scale with the number of features, leading to shorter training times for reduced feature sizes.
- *Memory usage*: A smaller feature vector requires less memory to store the feature values during training and classification processes. This can lead to reduced memory usage, which can provide cost-effectiveness if there are limitations on the available memory resources.
- *Computational complexity*: The computational complexity of the classification algorithms (SVM in the present case) can be influenced by the size of the feature vector. The computational complexity of SVM training and classification depends on the number of support vectors, which are the data points nearest to the decision boundary. The dimensionality of the problem decreases by reducing the number of features, and it becomes computationally less expensive to find the support vectors. Also, the number of kernel evaluations required during training and classification decreases, leading to faster execution.

Motivation

The state-of-the-art models could be categorized into two classes: (1) the models adopted a machine-learning approach, and (2) the models employed deep convolutional neural networks.

Machine learning typically involves the use of statistical models that are trained on labeled data to make predictions or decisions. Machine learning models are often simpler and more interpretable. These models can often be trained on smaller datasets with fewer parameters. The model's success significantly depends on the handcrafted features that are extracted from the data. Important concerns about handcrafted features are time consumption [25], the requirement of domain expertise and careful feature engineering [26], bias due to the designer's prior assumptions that may not capture all relevant information in the data, and limited scalability, generalization, and reproducibility due to problem-specific design. This can limit the effectiveness of the model and lead to suboptimal performance.

The deep convolutional neural networks address these concerns through their potential to auto-generate features from raw images. These networks are well known for producing human-like performance in the field of pattern recognition. The main

issues related to the implementation of these networks are the requirements of large datasets, millions of trainable parameters, and high computational complexity, which can restrict their deployment on low-end hardware platforms such as embedded systems, Raspberry Pi, field programmable gate arrays (FPGA), and cell phones.

The pros and cons of the abovementioned approaches induced the motivation for developing a recognition model to bridge the gap between them and receive optimum advantages.

Contribution

In the proposed work, the network architecture of benchmark DCNN models VGG-16Net, VGG-19Net, ResNet-50, and Inception-v3 was modified as a feature extractor to exploit their auto-generative feature capabilities. The classical PCA method was adopted for optimizing the size of feature vectors received from individual models. The optimized feature vectors were fused together in a strategic manner to obtain the maximum recognition rate from the benchmark SVM classifier. The suggested model provided a low-classification-cost solution to the proposed problem in terms of feature vector size.

Preliminary

An overview of the techniques used in the presented work is given in the following subsections.

VGG-16Net

This is a convolutional neural network architecture developed by the Visual Geometry Group (VGG) at Oxford University. It is named after the fact that it consists of 16 layers, which include convolutional layers, pooling layers, and fully connected layers [27]. The VGG-16Net architecture was designed for image recognition and classification tasks and achieved state-of-the-art performance on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014. The network consists of a series of 3×3 convolutional layers, each followed by a rectified linear unit (RELU) activation function and a 2×2 max pooling layer. The final layers of the network consist of fully connected layers that perform the classification task. VGG-16Net is a deep neural network that has 138 million parameters and requires significant computational resources to train. However, pre-trained versions of the network are available and can be used for transfer learning, which allows for faster training on new image recognition tasks.

VGG-19Net

VGG-19Net is a convolutional neural network architecture with 19 layers [27]. It was developed by the Visual Geometry Group (VGG) at the University of Oxford and achieved state-of-the-art performance in ILSVRC-2014. The architecture of VGG-19Net is like that of VGG-16Net but with the addition of three extra 3×3 convolutional layers. VGG-19Net has 143 million parameters and requires significant computational resources to train. However, pre-trained versions of the network are available and can be used for transfer learning, which allows for faster training on new image recognition tasks.

ResNet-50

ResNet-50 is a deep convolutional neural network architecture that was introduced by Microsoft Research in 2015. The name "ResNet" comes from "residual network," which refers to the use of residual connections, or skip connections, which allow information to bypass certain layers in the network. This helps mitigate the vanishing gradient problem, which can occur when training very deep neural networks [28]. ResNet-50 consists of 50 layers and is used primarily for image recognition tasks such as object detection and classification. The architecture of ResNet-50 is based on the building blocks known as residual blocks, which consist of two convolutional layers and a skip connection. The skip connection allows the input to be added directly to the output of the residual block, which helps preserve information and gradients through the network.

Inception-v3

Inception-v3 is a convolutional neural network architecture that was introduced by researchers at Google in 2015 [29]. The architecture of Inception-v3 is based on the use of "inception modules," which consist of several parallel convolutional layers with different filter sizes. This allows the network to capture features at multiple scales and helps reduce the computational cost of the network. Inception-v3 also uses a technique called "factorization," which decomposes large convolutions into smaller convolutions. This helps reduce the number of parameters in the network and improve its computational efficiency. Inception-v3 also includes other features such as batch normalization and dropout regularization, which enhance the generalization performance of the network.

Principal component analysis

It is a statistical method that can be used to reduce the dimensionality of a dataset by projecting the original data onto a lower-dimensional subspace defined by the principal components. This projection preserves as much of the original variability as possible while reducing the number of dimensions needed to represent the data [30]. PCA has several applications, including data compression, feature extraction, and the visualization of high-dimensional data. It is also commonly used as a preprocessing step for other machine learning algorithms to reduce the number of features and improve the accuracy of the model. The steps involved in the estimation of the principal components are described as follows:

Let X be a data matrix of dimension N × F, where N is the number of samples and F is the number of features.

1. Standardization of X:

$$Z = (X - \mu)/\sigma \tag{1}$$

where Z is the standardized data matrix, μ is the mean vector of X, and σ is the standard deviation vector of X. This transforms each feature of X to have zero mean and unit variance, which ensures that all features are on the same scale and have equal importance in the analysis.

2. Calculation of the covariance matrix related to standardized data:

$$S = (1/N) \times Z^T \times Z \tag{2}$$

where, S is covariance matrix and Z^T is the transpose of Z.

3. Determining the eigenvectors and eigenvalues of the covariance matrix by the following:where V and λ represent eigenvectors and eigenvalues respectively and can be denoted as follows:

$$S \times V = \lambda \times V \tag{3}$$

$$V = \left[V_1, V_2, V_3, \ldots V_F\right]$$

$$\lambda = \left[\lambda_1, \lambda_2, \lambda_3, \ldots \lambda_F\right]$$

The eigenvectors represent the principal components, and the eigenvalues represent the variance explained by each principal component.

4. Calculation of principal components:

$$PC = Z \times V \tag{4}$$

Where PC represents principal components.

Support vector machine

It is a popular and powerful machine learning algorithm used for classification and regression analysis. The basic idea behind an SVM is to find the hyperplane that best separates the data points of different classes. The hyperplane is chosen so that it maximizes the margin, which is the distance between the hyperplane and the closest data points in each class. The data points closest to the hyperplane are called support vectors. SVMs can handle both linearly separable and nonlinearly separable data by using different types of kernels. A kernel function transforms the original data into a higher-dimensional feature space, where it may become linearly separable. Some commonly used kernel functions include the linear, the polynomial, and the RBF kernels. In addition to binary classification, SVMs can be extended to handle multiclass classification problems by using techniques such as one-vs-all and one-vs-one [31]. SVMs have several advantages over other classification algorithms, including their ability to handle high-dimensional data, their robustness to overfitting, and their effectiveness even with small datasets. In the proposed work, an SVM classifier was employed with the one-versus-all technique and an RBF kernel. The classification cost of a one-versus-all SVM classifier can be calculated as follows:

Let "m" be the number of classes and "n" be the number of training samples. Let "d" be the dimensionality of the feature vector. During training, the one-versus-all SVM classifier trains m separate binary SVM classifiers, one for each class. Each binary SVM

classifier is trained on a subset of the training data that consists of the samples from one class and the samples from all other classes. Let C be the regularization parameter of the SVM, and let "k" be the kernel function used by the SVM. The training complexity of the one-versus-all SVM classifier can be expressed as follows:

$$O\left(m \times n^2 \times d\right) \times [\text{complexity of the kernel function k}] \tag{5}$$

During classification, the one-versus-all SVM classifier applies each of the m binary SVM classifiers to the test sample and selects the class with the highest score. Let "t" be the number of test samples. The classification complexity of the one-versus-all SVM classifier can be expressed as follows:

$$O\left(m \times t \times d\right) \times [\text{complexity of the kernel function k}] \tag{6}$$

The complexity of the RBF kernel (k) used in an SVM classifier depends on the number of training samples and the dimensionality of the feature vector. The RBF kernel function is defined as follows:

$$k\left(x, x^{'}\right) = \exp\left(-\Upsilon \times ||x - x^{'}||^2\right) \tag{7}$$

where x and $x^{'}$ are two feature vectors, $||.||$ is the Euclidean distance between them, and Υ is a parameter that determines the width of the kernel. The complexity of the RBF kernel function can be calculated as follows:

For a single evaluation of the kernel function, the time complexity is O(d), since we need to compute the Euclidean distance between the two feature vectors. To evaluate the kernel function for all pairs of training samples, the complexity is as follows:

$$O\left(n^2 \times d\right) \tag{8}$$

Since there are n^2 pairs of training samples and we need to compute the kernel function for each pair. From Eqs. (5), (6), and (8), it is obvious that the various complexities of the SVM classifier directly depend on the dimensionality (d) of the feature vector. This suggested that optimizing feature vectors in terms of dimensionality (size) would improve the classification cost. The same is true for other classifiers.

Methods

The complete overview of the proposed scheme is depicted in Fig. 2.

Input dataset

The input dataset is compiled from a public repository [5]. The dataset has accurate labelling for each handwritten numeral in Hindi script. The dataset exhibits a wide range of variations in writing styles, size, slant, stroke thickness, etc. that are commonly encountered in real-world scenarios. It has a balanced distribution of numerals across different classes, which can ensure bias-free training. The dataset has satisfactory sample counts of 20,000. All these reasons make it a suitable choice for the proposed work.

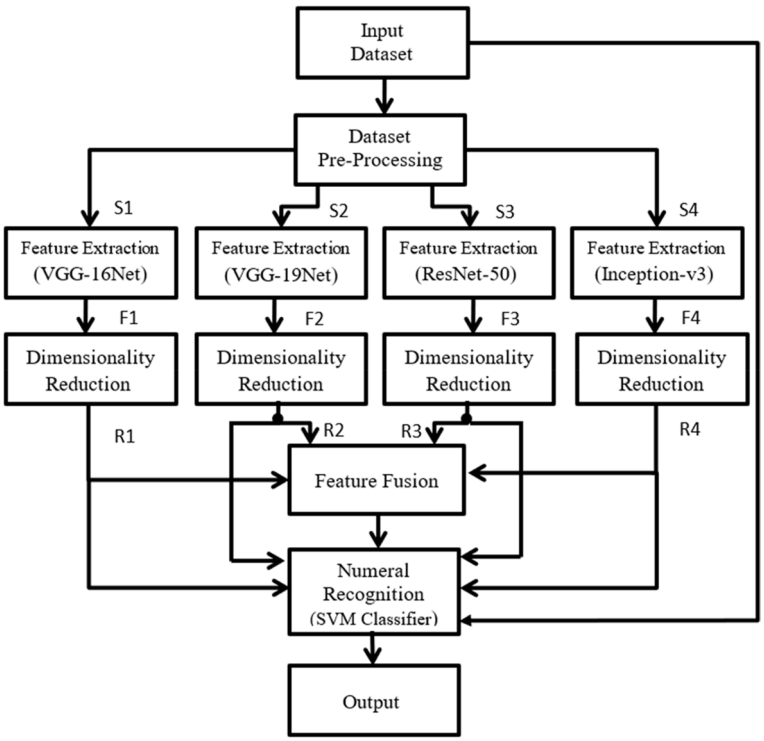

Fig. 2 Design of proposed model

Table 1 Input size requirements of DCNN models

Sl. no	DCNN model	Required input size	Resized images as per col. 3 were termed as
1	VGG-16Net	224 × 224 × 3	S1
2	VGG-19Net	224 × 224 × 3	S2
3	ResNet-50	224 × 224 × 3	S3
4	Inception-v3	299 × 299 × 3	S4

Dataset preprocessing

The pretrained DCNN models have specific input size requirements. In the presented work, images were resized in the input dataset to match the input size expected by the individual models. Details about the required input image size for proposed DCNN models are provided in Table 1.

The resized images for VGG-16Net, VGG-19Net, ResNet-50, and Inception-v3 were represented by S1, S2, S3, and S4, respectively, in Fig. 2.

Feature extraction

The architecture of individual models was modified for the purpose of feature extraction. The classification block of the individual DCNN models typically consists of fully connected layers with a large number of parameters (of the order of millions). These layers are responsible for mapping the extracted features to 1000 class labels, as the

classification blocks of individual models were originally designed to solve the classification problem of the ImageNet dataset with 1000 object classes. Since the objective of the proposed strategy is to exploit the auto-generative feature capability of pretrained DCNN models, their classification blocks are of no use. In the modified architecture, the classification blocks were removed completely to eliminate the computational burden and memory requirements associated with the fully connected layers. The remaining convolutional layers in the modified architecture were locked out of further training in order to take advantage of transfer learning. Arrangements have been made to collect the features after the final convolutional layer of each model. The individual models were set as feature extractors. The simplified architectures of modified networks are shown in Fig. 3. The resized images (S1, S2, S3, S4) were applied to the respective modified DCNN architectures, VGG-16Net, VGG-19Net, ResNet-50, and Inception-v3. The sizes of corresponding feature vectors derived for a given digit image were 4096, 4096, 2048, and 2048, respectively, and were represented by F1, F2, F3, and F4 in the design (refer to Fig. 2). The process of feature extraction is depicted in Algorithm 1.

Input: Resized images (S1, S2, S3, S4)
Output: Feature vectors (F1, F2, F3, F4)

1. For each DCNN model (VGG-16Ne, VGG-19Net, ResNet-50, Inception-v3)
   1.1 Modify the architecture for feature extraction
   1.2 Remove the classification section
   1.3 Lock the trainable layers from training
2. For each resized image (S1, S2, S3, S4)
   2.1 Apply the image to the respective modified DCNN architecture
   2.2 Collect the features at the output of the respective modified architecture
   2.3 Store the resulting features in the vector form
3. Return the feature vectors (F1, F2, F3, F4)

Algorithm 1. Algorithm for feature extraction

Feature optimization

This stage included the feature reduction and feature fusion steps of the proposed methodology. The background details of the numeral images were almost identical and did not carry any pattern-related information (refer to Fig. 1). This has suggested the possibility of having redundant information in the individual feature types (F1 to F4). The principal component analysis (refer to the "Inception-v3" section) has been applied to individual feature types F1 to F4 to eliminate feature collinearity. The trial-and-error technique was used to identify the optimum number of principal components. First, 10 PCA components were estimated using separate feature vectors (i.e., F1, F2, F3, and F4). These elements were combined to form one vector. A sample dataset of 500 such fused feature vectors (50 samples from each numeral class) was made for the specified purpose. The sample dataset was used to train and test the proposed classifier. For the sample dataset, the procedure was repeated while stepping up the principal component counts from 10 to 40 in increments of 2. The recognition accuracy was seen to greatly increase between components 10 and 20, but no further significant increases were seen. This suggests that 20 component counts are the optimal number. The various feature vectors (F1, F2, F3, and F4) were reduced in dimension by the suggested approach to 20, and the resulting reduced feature vectors were shown as R1, R2, R3, and R4 accordingly (refer to Fig. 2). The reduced features R1 to R4 were concatenated into a single feature vector Z. The frame

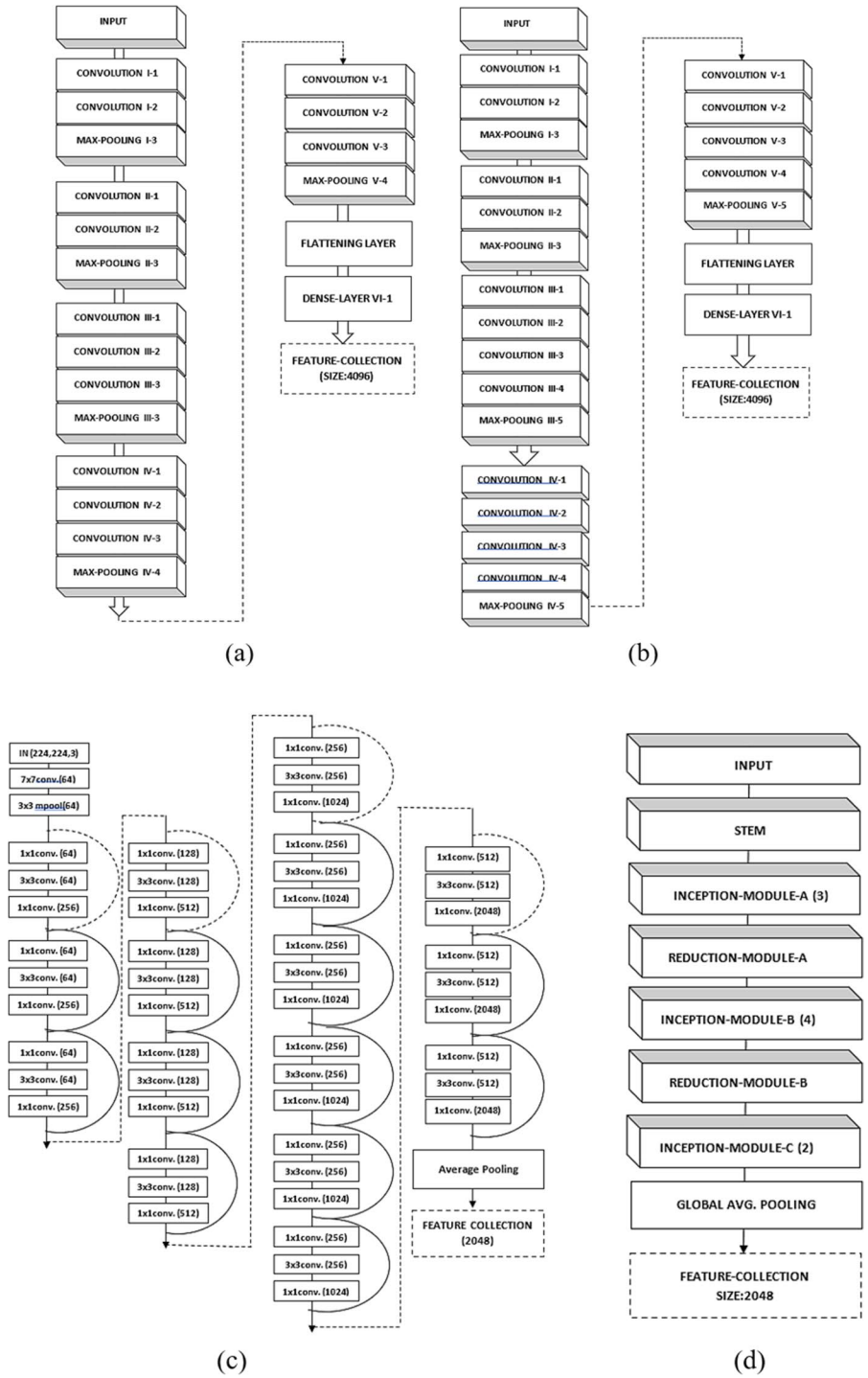

Fig. 3 Modified architecture of DCNN models as feature extractor. **a** VGG-16Net. **b** VGG-19Net. **c** ResNet-50. **d** Inception-v3

format of the fused feature vector Z is shown in Fig. 4. The size of the proposed optimized features becomes 80. The vector Z was estimated for all the numeral images in the input dataset. The reduced feature vectors R1 to R4 and the fused feature vector Z were used to create five new datasets. The process of feature optimization is depicted in Algorithm 2.

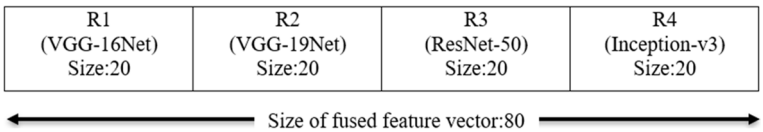

Fig. 4 Frame format of fused feature vector Z

Input: Feature vectors (F1, F2, F3, F4)
Output: Five new datasets (D1 to D5) including optimized features

1. For principal component counts from 10 to 40 in increments of 2
   1.1 Estimate 10 PCA components for each vector (F1, F2, F3, F4)
   1.2 Concatenate the PCA components into one vector
   1.3 Create a sample dataset of concatenated feature vectors for 500 samples (50 samples from each numeral class)
   1.4 Train and test the proposed classifier using sample dataset
   1.5 Record the recognition accuracy
2. Determine the optimum number of principal components based on the recognition accuracy results
3. For each feature vector (F1, F2, F3, F4)
   3.1 Reduce the dimensionality of each feature vector (F1, F2, F3, F4) to optimum number as received in step 2
   3.2 Store the resulting reduced feature vectors as R1, R2, R3, R4
   3.3 Concatenate the reduced feature vectors (R1 to R4) into a single vector Z
4. For each reduced feature vector (R1, R2, R3, R4)
   4.1 Create new dataset D1 by collecting R1 for all the 20,000 samples
   4.2 Create new dataset D2 by collecting R2 for all the 20,000 samples
   4.3 Create new dataset D3 by collecting R3 for all the 20,000 samples
   4.4 Create new dataset D4 by collecting R4 for all the 20,000 samples
5. Create new dataset D5 by collecting fused (optimized) feature vector Z for all the 20,000 samples

Algorithm 2. Algorithm for feature optimization

Numeral recognition

Besides the input dataset, five new datasets have been created up to this stage. The details are given in Table 2. Datasets D1 to D4 were created from the features received from VGG-16Net, VGG-19Net, ResNet-50, and Inception-v3, respectively, after feature optimization. Dataset D5 was created by concatenating the features related to datasets D1 to D4. The individual datasets were split into train and test sets in a ratio of 75:25. The SVM classifier was trained and tested with individual datasets. Details of the hyperparameters used during the classifier learning are given in Table 3. The results were recorded in terms of precision, recall, F1 score, and recognition accuracy. The formulations used for the calculation of the metrics are given in Table 4. Here, TP, TN, FP, and FN represent true-positive, true-negative,

Table 2 Summary of newly created datasets

Dataset	Originated from	Feature vector size	Size of dataset
Input	Pixel values of numeral images	1032	20,000
D1	VGG-16Net	20	20,000
D2	VGG-19Net	20	20,000
D3	ResNet-50	20	20,000
D4	Inception-v3	20	20,000
D5	Fusion of features related to D1, D2, D3, & D4	80	20,000

Table 3 Major parameters of SVM classifier

Parameter	Value
Kernel	RBF
Cache size	200 MB
Max iterations	− 1 (no limit)
Decision function shape	One versus all
Probability	False

Table 4 Details of performance metrics used in the proposed study

Metrics	Expression	Description
Recognition accuracy (A)	$\frac{TP+TN}{TP+FP+TN+FN}$	It counts the correct predictions out of total predictions
Precision (P)	$\frac{TP}{TP+FP}$	It counts the correct positive predictions out of the total positive predictions for given numeral class
Recall (R)	$\frac{TP}{TP+FN}$	It counts the correct positive predictions out of the total samples for given numeral class
F1 score (F1)	$2 \times \frac{P \times R}{P+R}$	It counts harmonic mean of precision and recall for given class

Table 5 Result obtained from input dataset (images)

Digit class	Precision	Recall	F1 score	Test samples
0	95.16	98.20	96.65	500
1	98.61	99.40	99.00	500
2	83.39	92.40	87.67	500
3	91.74	86.60	89.09	500
4	97.53	94.80	96.15	500
5	95.54	94.20	94.86	500
6	95.66	97.00	96.33	500
7	98.72	92.60	95.56	500
8	99.40	98.60	99.00	500
9	98.61	99.20	98.90	500
Overall recognition accuracy (%)		95.30		

false-positive, and false-negative events during the testing phase of the proposed classifier. A comprehensive result analysis is provided in the next section.

Results

Results obtained from various datasets are compiled in Tables 5, 6, 7, 8, 9 and 10. Arrangements have been made to estimate the confusion matrix using classification reports given in Tables 5, 6, 7, 8, 9 and 10. This would generate more readability about the model's performance. The consolidated results are compiled in Table 11. The model achieved highest recognition accuracy of 99.72% with the proposed fusion-based feature scheme (Dataset D5).

Table 6 Result obtained from dataset D1 (VGG-16Net)

Digit class	Precision	Recall	F1 score	Test samples
0	99.20	99.60	99.40	500
1	98.99	98.40	98.70	500
2	94.07	95.20	94.63	500
3	96.28	93.20	94.72	500
4	98.39	98.00	98.20	500
5	97.60	97.60	97.60	500
6	95.69	97.60	96.63	500
7	96.62	97.20	96.91	500
8	98.20	98.40	98.30	500
9	97.39	97.20	97.30	500
Overall recognition accuracy (%)		97.24		

Table 7 Result obtained from dataset D2 (VGG-19Net)

Digit class	Precision	Recall	F1 score	Test samples
0	99.60	99.60	99.60	500
1	97.60	97.80	97.70	500
2	94.29	92.40	93.33	500
3	94.25	95.00	94.62	500
4	98.02	99.20	98.61	500
5	98.59	97.80	98.19	500
6	95.83	96.60	96.22	500
7	98.41	98.80	98.60	500
8	99.20	98.60	98.90	500
9	96.00	96.00	96.00	500
Overall recognition accuracy (%)		97.18		

Table 8 Result obtained from dataset D3 (ResNet-50)

Digit class	Precision	Recall	F1 score	Test samples
0	100.00	99.60	99.80	500
1	98.59	98.00	98.29	500
2	94.72	96.80	95.75	500
3	96.95	95.40	96.17	500
4	97.43	98.60	98.01	500
5	97.80	97.60	97.70	500
6	95.10	97.00	96.04	500
7	98.60	98.40	98.50	500
8	98.40	98.20	98.30	500
9	97.55	95.40	96.46	500
Overall recognition accuracy (%)		97.50		

Feature separation

Arrangements were made to visualize the separation between the features related to various numeral classes in the input dataset and the proposed feature scheme (dataset D5) by using t-SNE (t-distributed stochastic neighbor embedding) algorithm. It is

Table 9 Result obtained from dataset D4 (Inception-v3)

Digit class	Precision	Recall	F1 score	Test samples
0	100.00	99.60	99.80	500
1	98.38	97.40	97.89	500
2	94.12	96.00	95.05	500
3	98.15	95.60	96.86	500
4	97.44	98.80	98.11	500
5	97.81	98.20	98.00	500
6	95.88	97.80	96.83	500
7	98.80	98.80	98.80	500
8	98.59	98.20	98.40	500
9	97.16	95.80	96.48	500
Overall recognition accuracy (%)		97.62		

Table 10 Result obtained from dataset D5 (proposed fusion-based features)

Digit class	Precision	Recall	F1 score	Test samples
0	100.00	100.00	100.00	500
1	100.00	99.80	99.90	500
2	98.23	99.80	99.01	500
3	99.80	98.80	99.30	500
4	99.40	100.00	99.70	500
5	99.80	99.60	99.70	500
6	100.00	99.80	99.90	500
7	100.00	100.00	100.00	500
8	100.00	99.80	99.90	500
9	100.00	99.60	99.80	500
Overall recognition accuracy (%)		99.72		

Table 11 Consolidated results obtained from various datasets used in the proposed study

Dataset	Recognition accuracy (%)
Input (images)	95.30
D1	97.24
D2	97.18
D3	97.50
D4	97.62
D5 (proposed)	99.72

a popular dimensionality reduction algorithm used for visualizing high-dimensional data in a low-dimensional space while preserving the structure of the original data as much as possible. With the help of Gaussian kernel, t-SNE computes a similarity score for each data point with every other data point based on Euclidean distance. The similarity scores were used to compute probability distributions for both the high-dimensional and low-dimensional spaces. The goal of t-SNE is to minimize the divergence between the probability distributions in the high-dimensional and low-dimensional

spaces. The algorithm does this by adjusting the positions of the data points in the low-dimensional space so that the probability distributions match as closely as possible.

A significant separation between the features related to different numeral classes could be observed in Fig. 6b, which derived from the proposed feature scheme, in comparison to Fig. 6a, which derived from the raw images of the input dataset. The more separation between the features, the easer their classification.

The results of various benchmark models, along with a proposed one, are compiled in Table 12. It should be noted that there is no standard dataset of handwritten Hindi numerals in the public domain, and the results of benchmark models as mentioned in Table 12 were based on different datasets.

The proposed model produced a comparable recognition rate to the benchmark models, that too with a smaller feature vector and a higher number of test samples. Small is the size of the feature vector, and low will be the training and classification complexities (refer to "Principal component analysis" section).

Table 12 Results of benchmark models along with proposed one

Sl. no	Benchmark model	Feature-types	Classifier	Dataset size	Test-sample count	Feature-vector size	Max. recg. acc. (%) for Hindi numerals
1	Khanduja et al. [3][a]	Hybrid of structural and statistical features (intersection points, end points, loops, and pixel distributions)	MLP	22,556	2000	462	95.5
2	Trivedi et al. [9]	Image	CNN with genetic algorithm and L-BFGS method	22,546	3762	256	96.54
3	Kumar et al. [10][b]	Image	Convolution autoencoder	17,000	3400	1032	99.59
4	Singh et al. [4]	Regional-weighted run length	MLP, NB, logistic, RF, SVM	6000	2000	196	95.02 with SVM
5	Chaurasiya et al. [11][b]	CNN-based features	SVM	22,556	3759	1600	99.41
6	Sarkhel et al. [12][b]	CNN-based features	SVM	3000	1000	4096, 2560, & 1792	99.5
7	Proposed model	Fusion-based features as received from VGG-16Net, VGG-19Net, ResNet-50, and Inception-v3	SVM	20,000	5000	80	99.72

[a] The benchmark model in [3] addressed the recognition problem of handwritten Devanagari characters and numerals individually. Recognition accuracy for numerals was included in the table to maintain relevance to the proposed study

[b] Benchmark models in [10–12] addressed the recognition problem of handwritten numerals related to various other scripts as well. Recognition accuracy for Devanagari numerals was included in the table to maintain relevance to the proposed study

Discussion

Figure 5 demonstrates the efficiency of the proposed scheme. The confusion matrix in Fig. 5a was derived when the classifier was tested with the input dataset (i.e., numeral images directly). A higher degree of confusion could be observed between numeral classes 2–3, 4–5, and 6–7; also, a significant count of false-negative (FN) predictions was

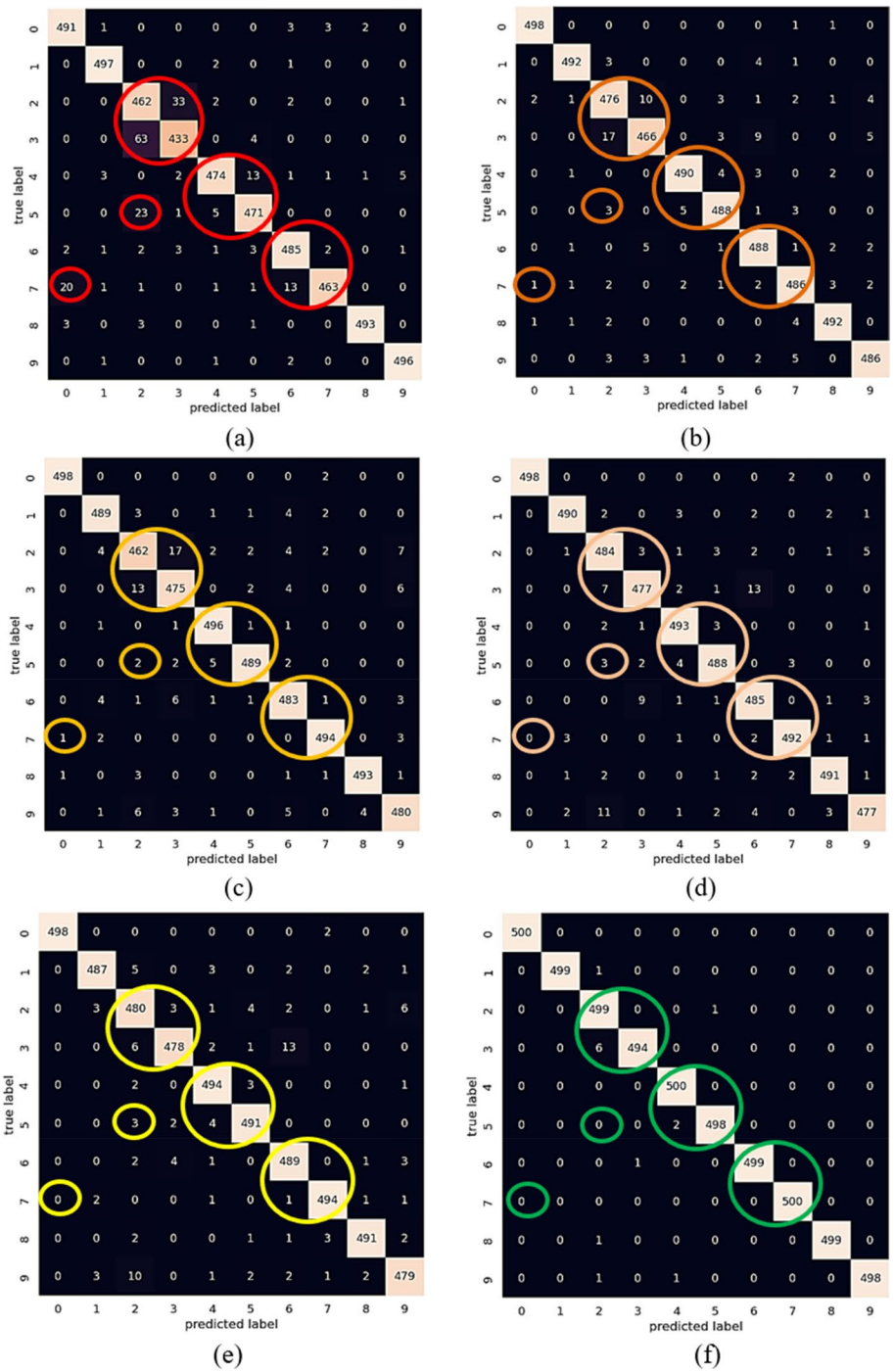

Fig. 5 Confusion matrix related to **a** input dataset, **b** dataset D1, **c** dataset D2, **d** dataset D3, **e** dataset D4, and **f** dataset D5 (proposed dataset)

recorded for numeral classes 5 and 7. All these regions of the confusion matrix were encircled in red. The confusion matrix in Fig. 5b to e was derived when the classifier was tested with datasets D1 to D4. Clear improvements could be observed in the encircled regions of the respective matrices with respect to Fig. 5a. This was also reflected in the recognition accuracy achieved with these datasets (refer to Table 11). The confusion matrix in Fig. 5f shows tremendous improvements over Fig. 5a and the rest. This matrix was derived by testing the classifier with the proposed fusion-based feature scheme (dataset D5). The matrix has minimal confusion. This suggested the potential of the proposed scheme in the selection of prominent features related to different numeral classes that could be helpful in their precise recognition by the given machine learning algorithm.

Figure 6 demonstrates the effectiveness of the proposed scheme in selecting distinct features related to various numeral classes. Proposed feature optimization resulted in a good separation between the features related to various numeral classes in the feature space, which contributed to achieve the comparable recognition rate to the benchmark models.

Referring to Table 12, the proposed model achieved comparable recognition accuracy to benchmark models by considering fewer numbers of features, which suggest its potential of solving the given problem with low-classification cost.

Conclusions

Most of the benchmark models relied on either a machine learning or deep learning approach. The former is simpler and more interpretable; it can be trained with small datasets and fewer parameters, but the need for manual feature engineering limits its performance. On the other hand, deep learning methods can autogenerate the salient features. These methods need large datasets and millions of trainable parameters to produce excellent results. The proposed study presented an effective ensemble of these state-of-the-art approaches. The benchmark DCNN models VGG-16Net, VGG-19Net,

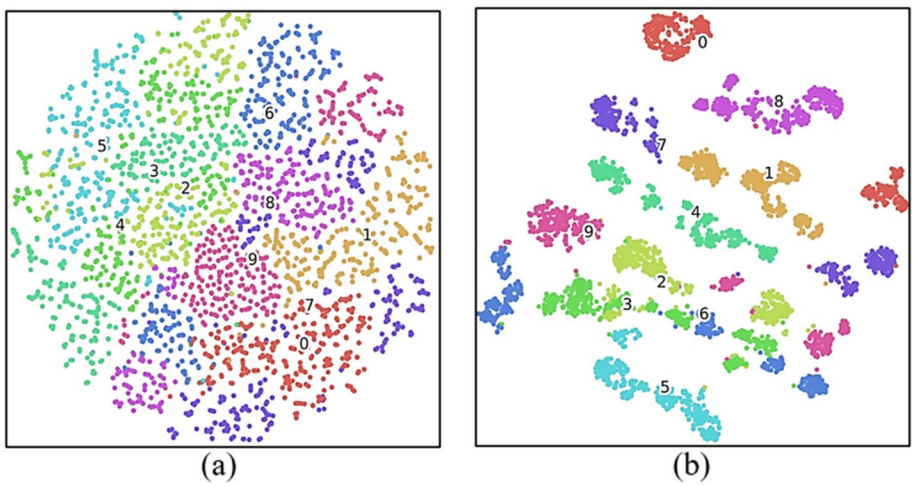

Fig. 6 Separation between the features related to various numeral classes in **a** input dataset and **b** proposed feature scheme (dataset D5)

ResNet-50, and Inception-v3 were employed as feature extractors that produced large feature vectors. The size of feature vectors was optimized by careful implementation of the classical PCA method, which led to a low-classification-cost solution to the proposed problem. The optimized features were fused together in a systematic manner and used to train the benchmark SVM classifier. The proposed model successfully achieved comparable results to the benchmark models with a smaller feature vector. Small is the size of the feature vector, and low will be the training and classification complexities. Although medical imaging and related pattern recognition problems are not within the scope of the current study, we are hopeful that the proposed fusion-based feature scheme would also be helpful in solving these kinds of problems effectively.

Abbreviations

PCA	Principal component analysis
SVM	Support vector machine
HOG	Histogram of gradient
FBNN	Feedforward backpropagation neural network
ELM	Extreme learning machine
MLP	Multilayer perceptron
NB	Naïve Bayes
RF	Random forest
DCNN	Deep convolutional neural networks
CNN	Convolutional neural network
BLSTM	Bidirectional long short-term memory
L-BFGS	Limited memory-Broyden–Fletcher–Goldfarb–Shanno
k-NN	K-nearest neighbor
RBF	Radial basis function
LSTM	Long short-term memory
FPGA	Field-programmable gate array
VGG	Visual Geometry Group
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
RELU	Rectified linear unit
TP	True positive
TN	True negative
FP	False positive
FN	False negative
t-SNE	T-distributed Stochastic Neighbor Embedding

Acknowledgements
We present our deep gratitude to Google Co-laboratory services to provide a hassle-free Python platform with the power of a graphical processing unit and vast python library support, without which it would be not easy to complete the proposed work. We are grateful to Acharya, Pant, and Gyawali for their efforts in developing the dataset of handwritten Devanagari characters and providing it in the public domain for progressive research in the related field.

Authors' contributions
All authors have equal contribution in the proposed research. All authors have read and approved the manuscript.

Funding
The proposed research does not involve any type of funding.

Availability of data and materials
The dataset used in the study is available on https://www.kaggle.com/datasets/ashokpant/devanagari-character-dataset-large

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 6 April 2023 Accepted: 2 July 2023
Published online: 20 July 2023

References

1. Das N et al (2012) A statistical-topological feature combination for recognition of handwritten numerals. Applied Soft Computing Journal 12(8):2486–2495
2. Iamsa-At S, Horata P (2013) Handwritten character recognition using histograms of oriented gradient features in deep learning of artificial neural network. International Conference on IT Convergence and Security, ICITCS-2013 1:1–5
3. Khanduja D, Nain N, Panwar S (2015) A hybrid feature extraction algorithm for Devanagari script. ACM Transactions on Asian and Low-Resource Language Information Processing 15(1):1–11
4. Singh PK, Das S, Sarkar R, Nasipuri M (2017) "Recognition of offline handwriten Devanagari numerals using regional weighted run length features," International Conference on Computer, Electrical and Communication Engineering, ICCECE-2016 1:1–6
5. Acharya S, Pant AK, Gyawali PK (2015) "Deep learning based large scale handwritten Devanagari character recognition," 9th International Conference on Software, Knowledge, Information Management and Applications, ICSKIMA-2015 9:1–6
6. Chakraborty B, Shaw B, Aich J, Bhattacharya U, Parui SK (2018) "Does deeper network lead to better accuracy: a case study on handwritten Devanagari characters," Proceedings - 13th International Workshop on Document Analysis Systems, DAS-2018 13:411–416
7. Sonawane PK, Shelke S (2018) "Handwritten Devanagari character classification using deep learning.," International Conference on Information, Communication, Engineering and Technology, ICICET-2018 1:1–4
8. Aneja N, Aneja S (2019) "Transfer learning using CNN for Handwritten Devanagari character recognition," 1st IEEE International Conference on Advances in Information Technology, ICAIT-2019 1:293–296
9. Trivedi A, Srivastava S, Mishra A, Shukla A, Tiwari R (2018) Hybrid evolutionary approach for Devanagari handwritten numeral recognition using convolutional neural network. Procedia Computer Science 125:525–532
10. S. Kumar and R. K. Aggarwal, "Augmented handwritten Devanagari digit recognition using convolutional autoencoder," International Conference on Inventive Research in Computing Applications, ICIRCA-2018. 2018:574–580.
11. S. Chaurasia and S. Agarwal, "Recognition of handwritten numerals of various Indian regional languages using deep learning," 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering, UPCON-2018. 2018:1–6.
12. Sarkhel R, Das N, Das A, Kundu M, Nasipuri M (2017) A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular Indic scripts. Pattern Recogn 71:78–93
13. Rakshit P, Chatterjee S, Haldar C, Sen S, Obaidullah SM, Roy K (2022) Comparative study on the performance of the state-of-the-art CNN models for handwritten Bangla character recognition. Multimedia Tools and applications 82(7):1–22
14. Garg A, Jindal MK, Singh A (2019) Offline handwritten Gurmukhi character recognition: k-NN vs. SVM classifier. Int J Inf Technol 13:2389–2396
15. Garg A, Jindal MK, Singh A (2019) Degraded offline handwritten Gurmukhi character recognition: study of various features and classifiers. Int J Inf Technol 14:145–153
16. Kathigi A, HonnamachanahalliKariputtaiah K (2022) Handwritten character recognition using skewed line segmentation method and long short term memory network. Int J Syst Assur Eng Manage 13(4):1733–1745
17. S. R. Narang, M. K. Jindal, S. Ahuja, and M. Kumar, "On the recognition of Devanagari ancient handwritten characters using SIFT and Gabor features," Soft Computing, no. published online, pp. 1–11, 2020.
18. Mushtaq F, Misgar MM, Kumar M, Khurana SS (2021) UrduDeepNet: offline handwritten Urdu character recognition using deep neural network. Neural Comput Appl 33:15229–15252
19. Raj MAR, Abirami S (2020) Junction point elimination based Tamil handwritten character recognition: an experimental analysis. J Syst Sci Syst Eng 29(1):100–123
20. Deore SP, Pravin A (2020) Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset. Sadhana - Acad Proc Eng Sci 45(1):1–13
21. Moudgil A, Singh S, Gautam V, Rani S, Shah SH (2023) Handwritten Devanagari manuscript characters recognition using CapsNet. Int J Cogn Comput Eng 4:47–54
22. H. Guo, Y. Liu, J. Zhao, and Y. Song, "Offline handwritten Tai Le character recognition using wavelet deep convolution features and ensemble deep variationally sparse Gaussian processes," Soft Computing, 2023.
23. Singh S, Garg N, Kumar M (2022) Feature extraction and classification techniques for handwritten Devanagari text recognition: a survey. Multimed Tools Appl 82:747–775
24. Jia W, Sun M, Lian J, Hou S (2022) Feature dimensionality reduction: a review. Complex Intell Syst 8(3):2663–2693
25. Janiesch C, Heinrich K. "Machine learning and deep learning". 2021:685–695.
26. Lecun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
27. Simonyan K, Zisserman A. "Very deep convolutional networks for large-scale image recognition". in 3rd International Conference on Learning Representations, ICLR-2015. 2015:1–14.
28. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-Dec:770–778
29. Szegedy C et al (2015) "Going deeper with convolutions," in IEEE Conference on Computer Vision and Pattern Recognition, CVPR-2015 24:1–9
30. Markos A, Tuzhilina E. "Principal component analysis," nature reviews methods primers. 2022;2.
31. Awad M, Khanna R (2015) Support Vector Machines for Classification. In: Efficient Learning Machines, vol 1. Apress, Berkeley, p 39–66

Danveer Rajpal received bachelor's degree in Electronics and Communication Engineering from University of Rajasthan, Jaipur, India, with Hons. in 2004; the master's degree in Digital Communication from JNU, Jodhpur, India, with Hons. and gold medal in 2011; and currently pursuing Ph.D. degree from Department of Electrical Engineering, Faculty of Engineering and Architecture, JNV University Jodhpur, India. He has served as Assistant Profesor in Department of Electronics and Communication in JECRC Engineering College, Jodhpur, India, from 2004 to 2012 and given his services as Head of the Dept. in Department of Electronics and Communication Engineering, VIET Engineering College, Jodhpur, India, from 2012 to 2019. His research interests include pattern recognition using machine learning and deep learning techniques.

Akhil Ranjan Garg received his BE (electrical engineering) and ME (control systems) degrees from M.B.M. Engineering College, Jodhpur, India, and subsequently did his Ph.D. from IIT Delhi, Delhi, India. He is presently working as Professor and Head in the Department of Electrical Engineering, Faculty of Engineering and Architecture, M.B.M. University Jodhpur, India. His research interests include computational neuroscience, intelligent systems, pattern recognition, power electronics, and machine learning. He has published over 40 papers in these areas in refereed international/national journals and conference proceedings. Dr. Garg is a recipient of numerous honors and awards including Young Teacher Career Award of AICTE, DAAD (German Academic Exchange Program) Fellowship, US Naval Academy Fellowship, and IBRO (International Brain Research Organization) Fellowship. He has served as Member of All India Board of Undergraduate Studies in Engineering and Technology (AIB-UGET) constituted by All India Council of Technical Education (AICTE), New Delhi; Member of Board of Governor IIT Jodhpur; Member Executive Council Central University of Rajasthan; Member Board of Management, Rajasthan Technical University Kota; and Member Academic Council MDS University Ajmer and Member Academic Council, JNV University, Jodhpur. He is former Honorary Chairman, Institution of Engineers (India) Jodhpur Local Centre. Dr. Garg is Fellow Institution of Engineers (India), Life Member Indian Society of Technical Education (ISTE), Member IEEE, Member International Brain Research Organization (IBRO), and Member International Neural Network Society (INNS).